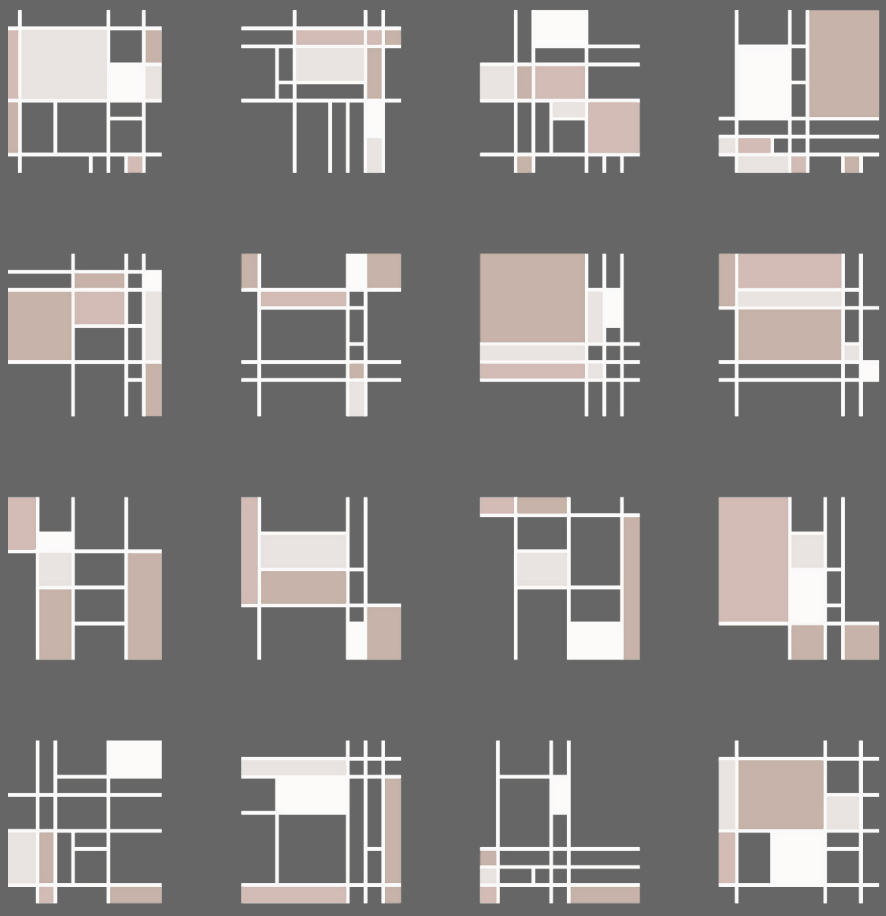
LA GRAMMATICA DEL PROGETTO

Shape grammar come strumento critico per la progettazione architettonica

Claudia Chirianni
tutor: prof. Roberta Amirante



Dottorato di ricerca in progettazione architettonica e urbana XXXII ciclo
Università degli Studi di Napoli Federico II - Dipartimento di Architettura

Abstract

La ricerca proposta ha come oggetto di studio *shape grammar*, con l'ambizione di raggiungere tre obiettivi:

- fare uno stato dell'arte delle ricerche condotte sul tema che possa costituire una base per ricerche future;
- stabilire quali siano i vantaggi offerti da questo metodo nell'avanzamento delle ricerche sulla progettazione architettonica rispetto ad altri metodi, computazionali e non;
- indagare i vantaggi che potrebbero derivare dall'ibridazione di shape grammar con tecniche di machine learning.

Introdotta da Stiny e Gips nel 1972[1], *shape grammar* è un sistema formale di calcolo basato su regole che produce forme geometriche e rappresenta uno dei primi tentativi di approccio computazionale all'architettura. Oggi è considerato uno dei sistemi generativi più interessanti per potenza e versatilità: si tratta infatti di uno straordinario strumento compositivo in grado di generare qualsiasi forma possibile. Il principale contributo della "grammatica di forma" alla progettazione è avvenuto attraverso una delle applicazioni di maggior successo dell'intelligenza artificiale, i *sistemi esperti*, capaci di emulare le capacità cognitive umane nel risolvere problemi che richiedono alti livelli di esperienza professionale. Con questo approccio sono state create numerose shape grammars, sia per l'analisi di opere o stili architettonici esistenti, sia per la realizzazione di progetti originali. Tra le applicazioni più interessanti del primo tipo vi è, ad esempio, la *Palladian Grammar*[2] di Stiny e Mitchell, in grado di "dedurre" le regole compositive di Palladio e di restituirne l'interno corpus di opere; mentre per il secondo tipo ricordiamo la *Malagueira Grammar*[3] di Duarte, che sistematizza le metodologie utilizzate da Siza per l'espansione di un insediamento residenziale a Malagueira. Tuttavia, oggi si stanno aprendo nuovi scenari, che la presente ricerca investiga, in cui shape grammar è utilizzata in combinazione con tecniche di machine learning e deep learning. Queste combinazioni permettono di utilizzare grammatiche più semplici, e soprattutto meno deterministiche rispetto ai sistemi esperti, e aprono una dimensione in cui la potenza "creativa" di shape-grammar può esprimersi con forza.

1 Stiny, G. & Gips, J. (1972). "Shape Grammars and the Generative Specification of Painting and Sculpture," in C. V. Freiman, ed., Information Processing 71, North Holland, Amsterdam, 1460-1465.

2 Stiny, G. & Mitchell, W.J. (1978). "The Palladian grammar," Environment and Planning B 5, 5-18.

3 Duarte, J.P. (2005). "Towards the mass customization of housing: the grammar of Siza's houses at Malagueira", Environment and Planning B: Planning and Design 32(3), 347-380.

La ricerca è strutturata in sei capitoli e presenta due casi studio a sostegno delle argomentazioni proposte.

Nel primo capitolo viene introdotta Shape Grammar, esplicitando i motivi alla base della scelta del tema di ricerca e gli obiettivi che questo studio si pone. Al fine di contestualizzarne il contributo specifico rispetto al panorama del design computazionale, vengono ripercorse le principali tappe della storia del design digitale, viene approfondito il contributo metodologico delle tecniche computazionali alla progettazione architettonica e, infine, viene esplorata la questione della creatività dei computer, della loro capacità di contribuire al processo creativo dei progettisti o perfino di produrre autonomamente soluzioni progettuali originali.

Nel capitolo 2 viene discussa shape grammar come sistema di produzione. Vengono dunque prima sottolineate analogie e differenze con la grammatica generativa chomskiana e i L-systems a cui si ispira, ne viene spiegato il funzionamento e infine viene illustrato come si manifesta l'emergenza.

Nel capitolo 3 è descritto l'uso di shape grammar come strumento di analisi, che verrà illustrato attraverso lo studio di due ricerche paradigmatiche, riportate in forma di schede di approfondimento: la Palladian Grammar di Stiny e Mitchell e il Prairie Language di Koning e Eisenberg.

Nel capitolo 4 vengono invece discusse le principali linee di ricerca sull'uso di shape grammar per la generazione di soluzioni progettuali originali. Nella scheda 3 viene approfondita la Discursive Grammar di J. Duarte (2001) per l'ampliamento del progetto di Siza a Malagueira, in quanto perfetto esempio di come soluzioni originali possano essere sviluppate a partire dall'analisi di opere esistenti. Nella scheda 4 viene invece approfondita CGA Shape, una grammatica sviluppata da Muller e Parish (2006) presso l'ETH di Zurigo, utilizzata per la creazione di un software, City Engine (2008), capace di generare soluzioni progettuali.

Nel capitolo 5 vengono analizzati approcci alternativi all'uso di shape grammar in cui le regole generative sono più semplici ("grammatiche ingenue") e il processo generativo viene guidato attraverso l'uso di tecniche di apprendimento automatico.

Vengono quindi presentati i due casi studio, in cui shape grammar viene messa a confronto con altri metodi di analisi di stili/opere architettonici al fine di valutarne l'efficacia.

Nel primo, viene fatto uno studio comparativo tra l'analisi di Eisenman e quella di Flemming di Casa Giuliani Frigerio, con lo scopo di evidenziare le differenze tra i due approcci e i vantaggi relativi all'uso di shape grammar come strumento di analisi.

Nel secondo caso studio, lo studio comparativo viene fatto tra shape grammar e altri metodi computazionali, utilizzati per l'analisi dello stile compositivo di Mondrian, al fine di comprendere vantaggi e limiti di ciascuno.

Infine, nel Capitolo 6, sulla base di tali considerazioni, saranno presentate le conclusioni del lavoro.

Ringraziamenti

Vorrei esprimere la mia più profonda gratitudine alla mia mentore, la prof.ssa Roberta Amirante: questo lavoro non sarebbe stato possibile senza il continuo stimolo intellettuale, la notevole pazienza e la fiducia dimostratami nel darmi assoluta libertà di perseguire le mie idee. Sono particolarmente grata inoltre alla prof.ssa Paola Scala per l'indispensabile supporto fornito durante il mio percorso.

Vedo questo lavoro come l'espressione delle capacità intellettuali che ho acquisito o rafforzato prendendo parte al corso di dottorato. Vorrei pertanto ringraziare i docenti del collegio del dottorato e i revisori della mia tesi, i cui consigli e osservazioni mi hanno permesso di migliorare e completare la tesi con una sensazione di grande soddisfazione.

Vorrei soprattutto ringraziare la mia famiglia, a cui dedico questo lavoro, per l'affetto incondizionato e il sostegno offertomi durante il periodo difficile in cui ho scritto la tesi, per aver sempre avuto fiducia in me e per aver fatto sempre tutto il possibile per aiutarmi a realizzare i miei sogni.

Indice

CAPITOLO 1

INTRODUZIONE

In questo capitolo introduttivo viene presentata Shape Grammar, esplicitandone il ruolo nella ricerca architettonica contemporanea.
Nel primo paragrafo se ne racconta l'origine e se ne dà una prima descrizione generale, sottolineando i motivi alla base della scelta del tema di ricerca e gli obiettivi che questo studio si pone. Nel secondo paragrafo, al fine di contestualizzarne il contributo specifico rispetto al panorama del design computazionale, vengono ripercorse le principali tappe della storia del design digitale sottolineando la relazione tra l'uso delle nuove tecnologie e l'introduzione, nella teoria architettonica, di concetti quali objectile, folding, non-linearità, etc. Nel terzo paragrafo viene dunque approfondito il contributo metodologico delle tecniche computazionali alla progettazione architettonica. Nell'ultimo paragrafo infine viene esplorata la questione della creatività dei computer, della loro capacità di contribuire al processo creativo dei progettisti o perfino di produrre autonomamente soluzioni progettuali originali.

1.1_Perche shape grammar?

Negli scorsi decenni gli studi sull'intelligenza artificiale hanno favorito lo sviluppo di nuovi strumenti per la progettazione architettonica tra cui shape grammar, un sistema basato su regole che genera forme geometriche[1]. Introdotta da George Stiny e James Gips con l'articolo seminale pubblicato nel 1972, "Shape Grammars and the Generative Specification of Painting and Sculpture"[2], shape grammar rappresenta uno dei primi sistemi generativi design-oriented e uno dei primi tentativi di approccio computazionale all'arte e all'architettura[3]. Cinque anni dopo, un altro testo di Stiny, "Two exercises in formal composition"[4], divenne il fondamento di molte applicazioni di shape grammar in architettura.

Fin dall'inizio infatti questo sistema ha riscosso grande successo in ambito accademico ed è stato oggetto di numerose ricerche e implementazioni. Tuttavia, qualche difficoltà legata al suo utilizzo hanno spinto alcuni ricercatori a cercare strade alternative per far fronte a proplemi che, pure, shape grammar è in grado di risolvere in maniera estremamente efficace. La scelta di shape grammar come tema di questa ricerca risiede proprio in questa contraddizione. Da un lato infatti la continua produzione di ricerche sul tema testimonia tanto il fatto che, a cinquant'anni dalla sua invenzione, l'interesse nei suoi confronti non sia ancora diminuito, quanto che il suo potenziale come strumento di progettazione generativa non è stato ancora pienamente sfruttato. Dall'altro, non si può non considerare che la costruzione di una shape grammar implica la codifica, al suo interno, dell'intero insieme di requisiti di progettazione e delle conoscenze e competenze di un progettista esperto. In altre parole, la costruzione di una buona grammatica progettuale richiede contemporaneamente competenze progettuali e di programmazione piuttosto avanzate. Tuttavia, come vedremo, shape grammar, come strumento compositivo "critico", risulta ancora molto vantaggiosa, se paragonata ad altri metodi computazionali, per qualità dei risultati ottenuti nella codifica di stili compositivi.

Attualmente shape grammar è poco conosciuta in Italia, pertanto un primo obiettivo di questa tesi è quello di fare uno stato dell'arte delle ricerche condotte sul tema

1 Kalay, Y.E. (2004) Architecture's New Media, Cambridge. MA: The MIT Press

2 Stiny, G. and Gips, J. (1972). "Shape Grammars and the Generative Specification of Painting and Sculpture," in C. V. Freiman, ed., Information Processing 71, North Holland, Amsterdam, 1460-1465.

3 Tepavčević, B., Stojaković, V. (2012). "Shape Grammar in contemporary architectural theory and design", Architecture and Civil Engineering Vol. 10, No 2, 169-178.

4 Stiny, G. (1976). "Two exercises in formal composition". Environment and Planning B: Planning and Design, 3(2), 187-210.

che possa costituire una base per ricerche future. Il secondo obiettivo è di stabilire quali siano i vantaggi offerti da questo metodo nell'avanzamento delle ricerche sulla progettazione architettonica rispetto ad altri metodi, computazionali e non. Il terzo obiettivo consiste nell'indagare i vantaggi che potrebbero derivare dall'ibridazione di shape grammar con metodi che ne permettano un uso meno difficoltoso.
Di seguito verranno illustrate le principali argomentazioni alla base di tali obiettivi.

Il funzionamento di shape grammar verrà ampiamente illustrato e discusso nei prossimi capitoli, tuttavia è utile anticipare alcune delle sue caratteristiche più tecniche in questa sezione per poter argomentare le ragioni alla base della scelta del tema di ricerca e gli obiettivi che questa si propone. Si rimanda alla lettura dei capitoli e paragrafi dedicati a ciascun argomento per descrizioni più approfondite.

Il contributo di shape grammar

Il principale contributo di shape grammar alla ricerca architettonica è avvenuto attraverso una delle applicazioni di maggior successo dell'intelligenza artificiale: i sistemi esperti.
Tali sistemi possono emulare le capacità cognitive umane nel risolvere problemi che si presume richiedano alti livelli di esperienza professionale in vari campi, dalla medicina alla progettazione[1]. I componenti caratteristici di un sistema esperto sono la base di conoscenza, che incorpora la conoscenza dell'esperto, e il motore inferenziale, il modulo che interpreta tale conoscenza per produrre una soluzione al problema. Il formalismo di rappresentazione e di elaborazione della conoscenza più diffuso nei sistemi esperti è quello delle "regole di produzione"; i sistemi formali di calcolo basati su di esse sono detti sistemi di produzione[2]. Nel loro insieme le shape grammars costituiscono una classe di sistemi di produzione finalizzati alla generazione e composizione di forme geometriche[3].

Come molti sistemi di produzione, le regole di shape grammar includono un lato sinistro (condizione) e un lato destro (conseguenza), ma tanto le regole quanto gli oggetti su cui operano sono costrutti geometrici.
In pratica, una shape grammar consiste in una forma iniziale, da cui far partire il calcolo, e un insieme di regole del tipo A→B, che sostituiscono la forma A con la forma B[1]. Dunque, il processo generativo di shape grammar si basa sulla riscrittura, un procedimento utilizzato nella grammatica generativa. Il sistema proposto da Gips e Stiny infatti è stato modellato su analogie linguistiche, in particolare sulla grammatica generativa di Chomsky di cui erano stati entrambi studenti al MIT. Tuttavia, a differenza della grammatica di riscrittura classica, i simboli di shape grammar vengono riscritti per rappresentare entità geometriche[2]:

"Where a phrase structure grammar is defined over an alphabet of symbols and generates languages of strings of symbols, shape grammars are defined over an alphabet of shapes and generate languages of shapes"[3].

Il vantaggio di shape grammar

Questo procedimento è alla base del successo delle shape grammars e della loro potenza generativa e versatilità: sono infatti in grado di produrre qualsiasi forma possibile[4]. D'altro canto, questo formalismo presenta vantaggi specifici e fornisce un metodo intuitivo per la definizione della forma che ha la capacità di generare soluzioni complesse e inaspettate a partire da poche e semplici regole[5]. Una delle caratteristiche fondamentali di shape grammar infatti è che permette l'emergenza[6], fenomeno che, come vedremo nel paragrafo 1.4, è alla base di qualunque processo creativo. Ciò fa sì che, come ha sottolineato Chase, i sistemi di progettazione basati su grammatiche abbiano un grande potenziale sia per automatizzare il processo di progettazione sia per consentire una vasta esplorazione di diverse alternative progettuali[7]. Sebbene infatti le forme generate da shape grammar presentino, come le frasi generate dalla grammatica testuale, delle somiglianze l'una con l'altra, essendo state generate dalle stesse regole compositive[8], al tempo stesso tali forme

1 Fox, J. (1996). "Expert systems and theories of knowledge", in M. Boden (ed.), Artificial Intelligence, Academic Press, 157-181.
2 Ibid.; Partridge, D. (1996). "Representation of Knowledge", in M. Boden (ed.), Artificial Intelligence, Academic Press, 55- 87; Enciclopedia Treccani: http://www.treccani.it/enciclopedia/sistemi-esperti. (visitato il 17/08/2020).
3 Kalay, Y.E. (2004) Architecture's New Media, Cambridge. MA: The MIT Press

1 Una descrizione dettagliata del formalismo di shape grammar verrà data nel capitolo 2.
2 Per un approfondimento sui sistemi di riscrittura e la differenza tra grammatica di Chomsky e shape grammar vedi capitolo 2, paragrafo 2.1.
3 Stiny, G. (1975). "Pictorial and Formal Aspects of Shape and Shape Grammars: On Computer Generation of Aesthetic Objects", Interdisciplinary Systems Research series 13, Basel and Stuttgart: Birkhäuser, 28. Traduzione: "Laddove la grammatica della struttura di una frase è definita su un alfabeto di simboli e genera linguaggi di stringhe di simboli, le grammatiche di forma sono definite su un alfabeto di forme e generano linguaggi di forme".
4 Stiny, G. (1975). Pictorial and Formal Aspects of Shape and Shape Grammars and Aesthetic System, Birkhauser, Basel.
5 Rowe, P.G. (1987). Design Thinking, MIT Press, Cambridge, MA.
6 Cfr. capitolo 2, paragrafo 2.3.
7 Chase, S.C. (2002). "A model for user interaction in grammar-based design systems", Automation in Construction 11, 161–172.
8 Kalay, Architecture's New Media, 274.

presentano anche una grande varietà, essendo trattate come entità non atomiche: possono cioè essere liberamente scomposte e ricomposte a discrezione del progettista[1]. Un vantaggio generativo di shape grammar rispetto ad altri metodi di sintesi formale deriva infatti dalla sua natura non deterministica: a ogni step del processo il progettista può scegliere di applicare una qualsiasi delle regole a qualsiasi parte corrispondente della forma[2].

I limiti di shape grammar

Poiché shape grammar codifica le descrizioni di forme che appartengono a uno stesso "linguaggio", può essere usata tanto come strumento di analisi, per descrivere forme appartenenti al corpus da cui sono derivate, quanto come strumento progettuale, per generare nuove forme appartenenti alla stessa "famiglia"[3]. Sebbene implicito nei primi lavori pubblicati da Stiny and Gips[4], l'uso della grammatica per la generazione di soluzioni inedite è stato esplicitamente affrontato per la prima volta solo nel 1980 da Stiny nel suo articolo "Kindergarten grammars: designing with Froebel's building gift"[5], dove esamina il metodo pedagogico di Frederick Froebel e la sua analogia con lo studio della composizione, proponendone un'interpretazione computazionale[6]. Lo studio di Stiny è stato poi implementato da Knight che ha evidenziato alcune difficoltà nell'uso della grammatica per la progettazione, relative alla traduzione di forme astratte e sperimentali in progetti architettonici che si adattino a contesti o programmi particolari[7].

"Different approaches to connecting grammars and goals have been suggested. One approach is direct. It involves writing rules with the foreknowledge that the generated designs will meet, or start to meet, given goals. In order to do this, the behaviors and outcomes of rules must be predictable in some way"[1].

Queste grammatiche "esperte", che includono una progettazione molto esperta e ben codificata all'interno delle loro regole, portano facilmente a progetti realizzabili, in cui la fattibilità è garantita dalla struttura stessa della grammatica[2]. Tuttavia, richiedono uno sforzo di ingegneria della conoscenza considerevole[3]. Le shape grammars esperte possono infatti diventare molto difficili da creare e modificare:

"[. . .] a mixture of knowledge types, together with the lack of adequate justifications of the different rules, makes the maintenance of such knowledge bases very difficult and time consuming"[4].

Inoltre, i meccanismi più comuni per introdurre le conoscenze di esperti all'interno delle regole promuovono l'uso di grammatiche deterministiche che tendono a produrre forme conosciute a priori[5].

Nuove prospettive

Sembra dunque necessario mediare tra la capacità generativa di shape grammar e la fattibilità delle soluzioni prodotte[6]. In una grammatica esperta correttamente formulata, qualsiasi esecuzione delle sue regole produrrà idealmente progetti realizzabili, poiché il progettista avrà previsto in qualche modo la natura di questi risultati; tuttavia, se sacrifichiamo la capacità divergente di shape grammar per perseguire la

1 Ibid.

2 Kalay, Architecture's New Media, 274.

3 Ibid. Cfr capitoli 3 (Shape Grammar come strumento di analisi) e 4 (Shape Grammar come strumento di progettazione).

4 Vedi ad esempio: G. Stiny and J. Gips, "Shape Grammars and the Generative Specification of Painting and Sculpture," in C. V. Freiman, ed., Information Processing 71 (North Holland, Amsterdam, 1972), pp. 1460-1465; Gips, J. (1975). Shape Grammars and their Uses: Artificial Perception, Shape Generation and Computer Aesthetics (Interdisciplinary Systems Research series 10). Basel and Stuttgart: Birkhäuser Verlag; Stiny, G. (1975). Pictorial and Formal Aspects of Shape and Shape Grammars: On Computer Generation of Aesthetic Objects (Interdisciplinary Systems Research series 13). Basel and Stuttgart: Birkhäuser; Stiny, G. and J. Gips (1978). Algorithmic Aesthetics, University of California Press, Berkeley, CA.

5 Stiny, G. (1980). "Kindergarten grammars: designing with Froebel's building gifts," Environment and Planning B 3: 409-462.

6 Ibid.

7 Knight, T.W. (1992). "Designing with grammars", in Schmitt G N (ed) Computer-Aided Architectural Design, Wiesbaden: Verlag Viewag, 33-48: 48.

1 Knight, T.W. (1999). "Applications in Architectural Design, and Education and Practice", Technical Report, NSF/MIT Workshop on Shape Computation, p.7. Traduzione: "Sono stati suggeriti diversi approcci per collegare grammatiche e obiettivi. Un approccio è diretto. Implica la scrittura delle regole con la consapevolezza che i progetti generati raggiungeranno o inizieranno a raggiungere determinati obiettivi. Per fare ciò, i comportamenti e gli esiti delle regole devono essere prevedibili in qualche modo".
Vedi anche: Knight, T.W. (1999). "Shape grammars: six types," Environment and Planning B: Planning and Design 26 (1), 15-31; Knight, T.W. (1999). "Shape grammars: five questions". Environment and Planning B: planning and Design, 26(4), 477-501.

2 Manuela Ruiz-Montiel et al., (2013) "Design with shape grammars and reinforcement learning". 232

3 Ibid.

4 Studer R., Benjamins V.R., Fensel D. (1998). "Knowledge engineering: principles and methods", Data and Knowledge Engineering 25, 161–197. Traduzione: "[. . .] una combinazione di tipi di conoscenza, insieme alla mancanza di adeguate giustificazioni delle diverse regole, rende molto difficile il mantenimento di tali basi di conoscenza e richiede tempo".

5 Manuela Ruiz-Montiel et al., (2013) "Design with shape grammars and reinforcement learning". 232.

6 Ibid.

prevedibilità, rischiamo di perdere uno dei motivi principali per usare questo formalismo, ovvero la possibilità di ottenere molte soluzioni impreviste e innovative[1]. Pertanto, lo sviluppo di metodologie adeguate alla creazione e il controllo delle shape grammars resta una questione aperta della ricerca architettonica.

Un approccio alternativo prevede l'uso di grammatiche più semplici e generiche in combinazione con tecniche di machine learning[2]. Una grammatica più semplice infatti risulterebbe troppo "debole" e potrebbe portare alla generazione di un numero illimitato di configurazioni formali inutili. Come per altri metodi generativi "meccanici", l'utilità di shape grammar per la sintesi progettuale dipende fortemente dal riconoscimento di forme "interessanti", più che dalla capacità del computer di generarle[3]. Attraverso l'uso di tali tecniche la generazione di forme può essere guidata verso soluzioni ottimali senza dover costruire sistemi di regole troppo articolate e specifiche. In questo senso è già stata fatta molta ricerca, soprattutto sulla combinazione con tecniche di ottimizzazione[4], tuttavia riteniamo che vi sia ancora un territorio molto vasto da esplorare per quanto riguarda la combinazione con tecniche di deep learning. La questione verrà approfondita nel capitolo 5.

1 Ibid.

2 Cfr capitolo 5.

3 Kalay, Architecture's New Media, 274-275.

4 J. Cagan, W.J.Mitchell (1993). "Optimally directed shape generation by shape annealing", Environment and Planning B: Planning and Design 20, 5–12; K. Shea, J.Cagan (1997). "Innovative dome design: applying geodesic patterns with shape annealing", Artificial intelligence for engineering design, analysis and manufacturing 11,379–394; K. Shea, J.Cagan (1999). "Languages and semantics of grammatical discrete structures", Artificial Intelligence for Engineering Design, Analysis and Manufacturing 13, 241–251; K. Shea, J.Cagan (1999). "The design of novel roof trusses with shape annealing: assessing the ability of a computational method in aiding structural designers with varying design intent", Design Studies 20, 3–23; J.S. Gero, S.J.Louis, S.Kundu (1994). "Evolutionary learning of novel grammars for design improvement", Artificial Intelligence for Engineering Design, Analysis and Manufacturing 8, 83–94; J.S. Gero, V.A.Kazakov (1996). "Evolving building blocks for design using genetic engineering: a formal approach", in: Advances in Formal Design Methods for CAD, Hall, pp.31–50; M.C. Ang, H.H.Chau, A.Mckay, A.de Pennington, "Combining evolutionary algorithms and shape grammars to generate branded product design", in: J.S. Gero (Ed.), Design Computing and Cognition '06,Springer, Netherlands, Dordrecht, pp.521–539; H.C. Lee, M.X.Tang (2009). "Evolving product form designs using parametric shape grammars integrated with genetic programming", Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23, 131–158; O. Chouchoulas (2003). Shape Evolution.An Algorithmic Method for Conceptual Architectural Design Combining Shape Grammars and Genetic Algorithms, Ph.D. Thesis, University of Bath; M. O'Neill, J.McDermott, J.M.Swafford, J.Byrne, E.Hemberg, A.Brabazon, E. Shotton, C.McNally, M.Hemberg (2010). "Evolutionary design using grammatical evolution and shape grammars: designing a shelter", International Journal of Design Engineering 3.

1.2_La svolta digitale in architettura

Come abbiamo visto nel paragrafo precedente, shape grammar rappresenta uno dei primi tentativi di approccio computazionale alla progettazione architettonica e, come altre applicazioni dell'intelligenza artificiale all'architettura, il suo successo è stato a lungo confinato agli ambienti accademici.

Possiamo dire, in generale, che l'interesse stesso nell'uso del computer per la progettazione architettonica inizia nelle università[1] e che questo strumento inizia ad entrare a pieno titolo nella pratica professionale solo a partire dagli anni Ottanta, grazie alla diffusione dei software CAD.

Per comprendere appieno il ruolo di shape grammar alla teoria e pratica progettuale in architettura ci sembra opportuno ripercorrere, in questo paragrafo, le principali tappe dello sviluppo del design computazionale, al fine di collocarla correttamente in questo vasto e variegato scenario. Nei prossimi paragrafi di questo capitolo verranno quindi presi in esame la relazione tra metodi computazionali e metodologia progettuale e il contributo che tali metodi possono dare alla progettazione come processo creativo.

Fig. 1.2_1. Ivan E. Sutherland al lavoro con il suo sistema di disegno Sketchpad, c.a. 1963. (Blau, Eve, Kaufman, Edward (eds.) (1989), Architecture and its Image. Four Centuries of Architectural Representation)

Le origini

Nel 1963 Ivan Sutherland presenta, presso il Massachusetts Institute of Technology, Sketchpad[2], il primo programma di disegno interattivo assistito da computer. Il sistema, che prevedeva una penna luminosa come dispositivo di input e un oscilloscopio modificato come dispositivo di output[3], utilizzava già, nella sua formulazione tridimensionale, alcuni schemi sintetici di volumi di edifici, lasciando intuire la potenzialità di un sistema integrato di controllo della fase progettuale[4]. Sutherland

1 Kalay, Y. E. (2004). Architecture's new media: Principles, theories, and methods of computer-aided design. MIT press, 66. Vedi anche: Dawson, J. W. (1961). "The Computer in Building Design", Architectural and Engineering News, vol. 3(12), 14–19; Eberhard, J. P. (1962). "A Computer-Based Building Process: Its Potentials for Architecture", Architectural and Engineering News, vol. 4(12), 16–18.

2 Sutherland, I. E. (1963), Sketchpad. A Man-Machine Graphical Communication System, Ph.D. Thesis, Massachusetts Institute of Tecnology, Cambridge, Mass.

3 Salomon, D. (2011). The Computer Graphics Manual, Texts in Computer Science, David Gries and Fred. B. Schneider (ed.), vol. 2, London: Springer Science & Business Media, 10; Marinčić, N. (2017), Towards Communication In Caad: Spectral Characterisation and Modelling with Conjugate Symbolic Domains, PhD Thesis, ETH Zurich (pubblicato da Birkhauser nel 2019).

4 Sdegno A (2016). "Computer Aided Architecture: origini e sviluppo", DISEGNARECON volume 9, n.16, 4.1-4.6: 4.2; Vedi anche: Sdegno, A. (2013) "Sketchpad: sulla nascita del disegno digitale", Disegnare Idee Immagini, 46, XXIII, 74-81; Sdegno, A. (2009). "Breve storia della rappresentazione numerica", in R. Migliari, Geometria descrittiva, Volume I – Metodi e costruzioni, CittàStudi-De Agostini, Novara, 245-252.

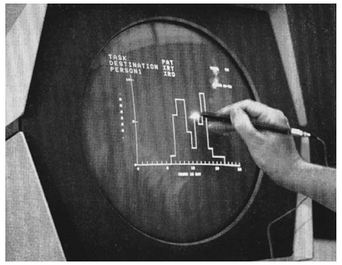


Fig. 1.2_2. Diagrammi di analisi con il software COPLANNER. (Souder, J. J. et al. (1964). Planning for Hospitals. A System Approach Using Computer- Aided Techniques)

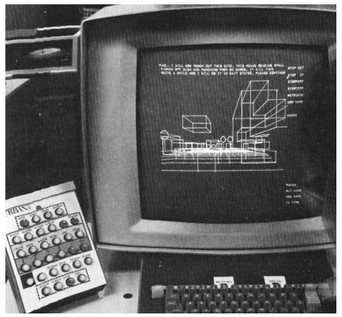


Fig. 1.2_3. Architecture Machine Group, Sistema di controllo e gestione URBAN5. (Negroponte, N. (1970). The Architecture Machine).

e il suo mentore, Steven A. Coons, inventano pertanto sia il moderno concetto di computer-aided design sia lo strumento per implementarlo[1]. Quest'idea viene riassunta da Coons nel paper presentato nel 1963 al congresso AFIPS:

"We envisioned even then the designer seated at a console, drawing a sketch of his proposed device on the screen of an oscilloscope tube with a "light pen," modifying his sketch at will, and commanding the computer slave to refine the sketch into a perfect drawing, to perform various numerical analyses (…)In some cases the human operator might initiate an optimization procedure to be carried out entirely automatically by the computer (…)"[2].

L'approccio di Sutherland fu emulato da Don Hart e Ed Jacks alla General Motors che, in collaborazione con IBM, sviluppa il sistema DAC-1 nel 1964 e, verso la fine degli anni '60, sistemi simili erano utilizzati in molte aziende automobilistiche e aerospaziali[3]. Per quanto riguarda l'architettura invece, come già accennato, l'interesse per l'uso del computer per la progettazione inizia in ambito accademico. In particolare, acquista notevolmente credito a seguito della pubblicazione di Notes on the Synthesis of Form[4] di Alexander, nel 1964, in cui viene descritto l'uso sistematico di un metodo computazionale per la progettazione architettonica[5]. Attorno a questo tema vengono organizzate in quegli anni varie conferenze (ad esempio, al Boston Architectural Centre nel 1964 e a Yale nel 1968) e vengono sviluppati vari sistemi di progettazione assistita al computer come, ad esempio, URBAN5[6], un sistema di progettazione urbana assistita al computer progettato da Negroponte e Groisser nel 1964, e COPLANNER[7] un metodo di Computer-Aided Planning basato sull'analisi funzionale-distributiva degli ospedali[8].

Cfr. il documentario Computer Sketchpad della National Education Television, MIT 1964, relativo al sistema di disegno Sketchpad III di Timothy Johnson.

1 Kalay, Architecture's New Media, 65.

2 Coons, S. A. (1963, May). An outline of the requirements for a computer-aided design system. In Proceedings of the May 21-23, 1963, AFIPS spring joint computer conference,299-304: 299.

3 Kalay, Architecture's New Media, 66.

4 Alexander, C. (1964). Notes on the Synthesis of Form. Harvard University Press.

5 Kalay, Architecture's New Media, 66.

6 Si veda il capitolo intitolato "URBAN5", in: Negroponte, N. (1970). The Architecture Machine, The MIT Press, Cambridge, 70-93.

7 Sounder, J. J. And Clark, W. E. (1963). "Computer Technology: A New Tool for Planning". AIA Journal, October: 97–106; Souder, J. J., Clark, W. E., Elkind, J. I., Brown, M. B. (1964). Planning for Hospitals. A System Approach Using Computer- Aided Techniques, American Hospital Association, Chicago.

8 Sdegno, A (2016). "Computer Aided Architecture: origini e sviluppo", 4.2

La prima generazione di sistemi CAD

Ancora negli anni Settanta i sistemi sviluppati dall'industria erano più orientati ad assistere gli ingegneri nella modellazione di geometrie complesse, mentre le prime ricerche esplicitamente rivolte alla definizione di sistemi computazionali per l'architettura veniva condotta principalmente nelle università americane e inglesi[1]. Negli Stati Uniti la ricerca condotta presso l'Institute of Physical Planning della Carnegie-Mellon University dal gruppo diretto da Charles M. Eastman, era centrata sullo sviluppo di sistemi di descrizione degli edifici e progettazione spaziale capaci di supportare, in un unico pacchetto e usando un solo database, tutte le operazioni relative alla costruzione[2]. Da queste ipotesi nacque nel 1974 il Building Description System[3] (BDS), da cui prese piede la sperimentazione che condusse all'attuale definizione del protocollo BIM (Building Information Modeling)[4]. Il BDS venne poi ampliato e modificato nel 1977 nel Graphical Language for Interactive Design (GLIDE), basato su un linguaggio di programmazione interpretato e capace di supportare operazioni geometriche su descrizioni parametriche di edifici[5]. Nello stesso periodo, un gruppo di ricercatori dell'Università del Michigan diretto da Harold Borkin, sviluppò il Computer Aided Engineering and Architectural Design System (CAEADS), un sistema di progettazione capace di supportare l'analisi energetica, di abitabilità e di verifica delle specifiche di costruzione[6]. Presso il MIT, l'Architecture Machine Group, fondato da Negroponte nel 1967, sperimentò un approccio basato sull'applicazione dell'intelligenza artificiale alla progettazione architettonica. Nel libro The Architecture Machine[7], Negroponte prevedeva il futuro del design come collaborazione tra designer umani e macchine: una simbiosi di due specie intelligenti[8]. Cinque anni dopo, in Soft Architecture Machines[9], Negroponte integra l'idea cibernetica di feedback nel suo lavoro[10].

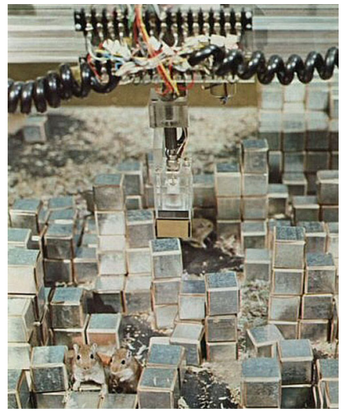


Fig. 1.2_4. Il SEEK project di Negroponte: il progetto illustra l'idea di "architettura reattiva, all'interno di un ambiente intelligente, distaccato, autosufficiente, sostenibile e completamente controllato/regolato" proposta in Soft Architecture Machine. (Marinčić (2017), Towards Communication In Caad).

1 Kalay, Architecture's New Media, 67

2 Kalay, Architecture's New Media, 67

3 "Our premise was that a computer database could be developed that would allow the geometric, spatial, and property description of a very large number of physical elements, arranged in space and 'connected' as in an actual building. Conceptually, the model would be similar to a balsa wood model, but with far greater detail": Eastman, C. M., et al. (1974), "An Outline of the Building Description System", Research Report No. 50, Carnegie-Mellon Univ., Pittsburgs, Pa. Inst. Of Physical Planning.

4 Sdegno A (2016). "Computer Aided Architecture: origini e sviluppo", 4.2.

5 Kalay, Architecture's New Media, 67

6 Kalay, Architecture's New Media, 67

7 Negroponte, N. (1970). The Architecture Machine, The MIT Press, Cambridge.

8 Si veda il capitolo intitolato "Architect-Machine Symbiosis", in: Negroponte, N. (1972). The Architecture Machine, seconda edizione, The MIT Press, Cambridge, 8-29. Vedi anche: Kalay, Architecture's New Media, 67-68; Marinčić, Towards Communication In Caad, 49-51.

9 Negroponte, N. (1975), Soft Architecture Machines, The MIT Press, Cambridge, Mass.-London.

10 Marinčić, Towards Communication In Caad, 50.

Nel Regno Unito lo sviluppo di applicazioni computazionali relative all'edilizia ha interessato principalmente la progettazione di edifici pubblici di larga scala[1]. Tra le più rilevanti vi erano OXSYS e HARNESS (1970), per la progettazione di ospedali; CEDAR, per la progettazione di edifici postali; e SSHA (1969-73), per l'edilizia residenziale economica[2].

La rivoluzione CAD

Questi primi approcci "accademici" allo sviluppo di sistemi computazionali per la progettazione avevano il merito di nascere dal punto di vista della progettazione architettonica piuttosto che da quello dell'informatica[3]. Tuttavia le loro soluzioni, se da un lato tendevano a essere più aderenti al processo progettuale architettonico, dall'altro erano spesso troppo poco maneggevoli per essere usate nella pratica professionale o troppo specifiche per essere di uso generale[4]. In molti casi queste ricerche si intrecciavano con riflessioni sulla metodologia progettuale e sull'intelligenza artificiale[5] e hanno portato alla creazione di sistemi CAD knowledge-based come Worldview[6], KAAD (Knowledge-Assisted Architectural Design)[7], ICAAD[8], SEED[9] e Building Design Advisor (BDA)[10].

Fig. 1.2_5. AutoCAD Release 1 (https://medium.com).

Nessuno di questi sistemi tuttavia ha riscontrato un successo commerciale, per cui la diffusione popolare del CAD in architettura avvenne grazie allo sviluppo di sistemi di disegno e modellazione generici[11]. Tuttavia, i primi computer avevano costi proibitivi e solo con l'avvento dei personal computer, in particolare l'IBM PC nel 1981, i software CAD entrarono finalmente negli studi di architettura[12]. Aziende come Summagraphics, MicroStation e Autodesk iniziarono a creare software specializzati per supportare la progettazione architettonica. Nel 1982 Autodesk rilascia la prima versione di AutoCAD (AutoCAD Release 1) che ancora oggi rimane uno degli strumenti di disegno più popolari[1].

1 Kalay, Architecture's New Media, 68.
2 Ibid.
3 Ibid.
4 Ibid.
5 L'argomento verrà approfondito nel paragrafo 1.3.
6 Kalay, Y. E. (1987). Worldview: An integrated geometric-modeling/drafting system. IEEE computer graphics and applications, 7(2), 36-46.
7 Carrara, G., Kalay, Y. E., & Novembri, G. (1994). Knowledge-based computational support for architectural design. Automation in Construction, 3(2-3), 157-175.
8 Pohl, J., & Myers, L. (1994). A distributed cooperative model for architectural design. Automation in Construction, 3(2-3), 177-185.
9 Flemming, U. (1994). "Case-based design in the SEED system". Automation in construction, 3(2-3), 123-133.
10 Papamichael, K., LaPorta, J., & Chauvet, H. (1997). Building Design Advisor: automated integration of multiple simulation tools. Automation in construction, 6(4), 341-352.
11 Kalay, Architecture's New Media, 68.
12 Marinčić, Towards Communication In Caad, 34; vedi anche: Weisberg, "The Engineering Design Revolution." http://www.cadhistory.net., capitolo 8

Architettura della variazione

Secondo R. Aish, studioso dell'interazione uomo-macchina ed ex direttore dello sviluppo software di Autodesk, le applicazioni CAD di questo tipo operano a un "livello semantico" molto basso (in quanto lavorano con elementi quali linee, archi e cerchi) e, limitandosi a restituire graficamente i risultati del processo di progettazione, risultano insufficienti come strumenti di progettazione computazionale. D'altro canto, applicazioni che presentano un livello semantico superiore ma predefinito (pareti, finestre e porte) codificano convenzioni costruttive esistenti in strumenti di progettazione, "costringendo" il progettista a pensare in modo specifico e dettagliato al progetto prima ancora di definire il concept progettuale generale[2]. L'idea di Aish è che vi debba essere una corrispondenza tra strumenti di progettazione computazionale e pensiero progettuale e che tali strumenti debbano rispondere alle opportunità offerte da approcci più esplorativi alla progettazione architettonica[3]. Lo sviluppo del software dipende pertanto dalla definizione del pattern generale e ricorrente del processo progettuale, cioè la "creazione e manipolazione di relazioni geometriche"[4]:

"If we could formalize a system that understands geometric relationships in the same way a creative designer does, then this would be worth computerization, because it is with such a system that the designer can create and express his own architectural semantics"[5].

Questa idea di processo progettuale basato sulla definizione di elementi e delle loro relazioni reciproche è alla base del concetto di progettazione parametrica e di spazio topologico[6]: un processo di questo tipo consente l'esplorazione di molteplici

1 Marinčić, Towards Communication In Caad, 34; vedi anche: Weisberg, "The Engineering Design Revolution." http://www.cadhistory.net., capitoli 6 e 8.
2 Aish R. (2005). "Extensible computational design tools for exploratory architecture", in Kolarevic B. (a cura di), Architecture in the Digital Age: Design and Manifacturing, Routledge, capitolo 17; vedi anche: Pugnale, A. (2012). "Engineering Architecture: Come il virtuale si fa reale", Bloom, n.14.
3 Aish R. (2005). "Extensible computational design tools for exploratory architecture", 339.
4 Ibid, 342.
5 Ibid. Traduzione: "Se potessimo formalizzare un sistema che comprende le relazioni geometriche nello stesso modo di un designer creativo, allora varrebbe la pena informatizzare, perché è con un tale sistema che il designer può creare ed esprimere la propria semantica architettonica".
6 Vedi: Kolarevic B. (2003). "Digital Morphogenesis", in Kolarevic B. (a cura di), Architecture in the digital age: design and manufacturing, Spon Press, capitolo 2; Pugnale, A. (2012). "Engineering Architecture: Come il virtuale si fa reale", Bloom, n.14.

soluzioni progettuali, tanto che Lars Spuybroek, del gruppo NOX, definisce un'architettura così concepita come un'architettura della variazione[1].

Software parametrici

Il primo software parametrico di successo commerciale è stato Pro/ENGINEER, rilasciato dalla Parametric Technology Corporation nel 1988[2]. Durante un'intervista con Industry Week nel 1993, Samuel Geisberg, ex professore di matematica che aveva fondato l'azienda nel 1985, afferma:

"The goal is to create a system that would be flexible enough to encourage the engineer to easily consider a variety of designs. And the cost of making design changes ought to be as close to zero as possible. In addition, the traditional CAD/CAM software of the time unrealistically restricted low-cost changes to only the very front end of the design-engineering process"[3].

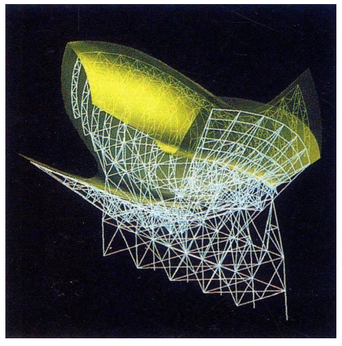


Fig. 1.2_6. Disegno CAD di Frank Gehry del 1993. (Marinčić (2017), Towards Communication In Caad).

Nel 1993 Dassault Systèmes ha incorporato molte delle funzioni parametriche di Pro/ENGINEER in CATIAv4[4]. All'epoca Gehry Partners collaborava con Rick Smith, un consulente IBM specializzato in CATIA, per realizzare progetti di architettura geometricamente complessi come il Barcelona Fish (1991) e il Guggenheim Museum di Bilbao (1993-97). Per l'ufficio di Gehry, un modello digitale creato in CATIA era la singola fonte di tutte le informazioni di progettazione, fabbricazione e costruzione dell'edificio[5]. Nel 2004 la società gemella di Gehry Partners, Gehry Technology (incorporata nel 2001) rilascia Digital Project, un pacchetto software basato su CATIA pensato espressamente per la progettazione architettonica. Grazie a componenti di costruzione parametricamente definiti, permetteva agli architetti di evolvere gradualmente i loro modelli da semplici schizzi bidimensionali a un modello tridimensionale elaborato e completamente dettagliato[1].

1 Spuybroek, L. (ed.) (2009). Research & Design: The Architecture of Variation, Thames & Hudson; vedi anche: Pugnale, A. (2012). "Engineering Architecture: Come il virtuale si fa reale", Bloom, n.14.

2 Weisberg, D. "The Engineering Design Revolution" http://www.cadhistory.net., capitolo 16

3 Geisberg citato in: Teresko, J. (1993). "Parametric Technology Corp.: Changing the way Products are Designed." Industry Week, December 20, 28. Traduzione: "L'obiettivo è creare un sistema sufficientemente flessibile da incoraggiare l'ingegnere a prendere facilmente in considerazione una varietà di progetti. E il costo per apportare modifiche al design dovrebbe essere il più vicino possibile allo zero. Inoltre, il tradizionale software CAD/CAM dell'epoca limitava irrealisticamente le modifiche a basso costo al solo front-end del processo di progettazione".

4 Weisberg, David. 2008. "The Engineering Design Revolution: The People, Companies and Computer Systems that Changed Forever the Practice of Engineering." http://www.cadhistory.net., capitolo 13.

5 Kolarevic, B. (2003). "Information Master Builders", in Kolarevic, B. (ed.) Architecture in the digital age: design and manufacturing, Spon Press, 88-96: 92

La necessità di un sistema per una descrizione adeguata degli edifici ai fini dell'industria delle costruzioni ha portato allo sviluppo di software per lo scambio di informazioni e la gestione delle attività di costruzione; questa famiglia di software è oggi conosciuta con il nome generico di Building Information Modeling (BIM)[2].

Dalle NURBS all'"architettura non standard"

Da un punto di vista prettamente geometrico, il termine parametrico viene utilizzato per definire "quelle curve e superfici utilizzate per rappresentare accuratamente forme libere, organiche o particolarmente complesse, cioè non riconducibili, se non con l'approssimazione, a geometrie semplici"[3]. A partire dagli anni '90 lo sviluppo tecnico dei modellatori di spline ha accompagnato l'emergere di una nuova tettonica digitale[4]:

"Three-dimensional digital modeling software based on NURBS (Non-Uniform Rational B-Splines), i.e. parametric curves and surfaces, has opened a universe of complex forms that were, until the appearance of CAD/CAM technologies, very difficult to conceive, develop and represent, let alone manufacture. A new formal universe in turn prompted a search for new tectonics that would make the new undulating, sinuous skins buildable (within reasonable budgets)"[5].

Grazie a queste nuove tecnologie gli architetti, ispirandosi agli scritti di Leibniz e Deleuze, hanno dunque iniziato a esplorare geometrie non euclidee e spazi topologici[6], arrivando a concepire forme fino ad allora impensabili come quelle dei

1 Marinčić, Towards Communication In Caad, 65

2 Ibid., 66; Kalay, Architecture's New Media, 67; Carpo, M. (2013). "Twenty Years of Digital Design", in Carpo, M. (ed). The Digital Turn in Architecture 1992-2012, John Wiley & Sons , 12. Per un approfondimento sulla tecnologia BIM vedi: Eastman C., Teicholz P., Sacks R., Liston K. (2008). BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors, Wiley.

3 Pugnale, A. (2012). "Engineering Architecture: Come il virtuale si fa reale", Bloom, n.14, 19.

4 Carpo, M. (2013). "Twenty Years of Digital Design", 9.

5 Kolarevic, B., Architecture in the digital age: design and manufacturing, 7. Traduzione: "I software di modellazione digitale tridimensionale basati su NURBS (Non-Uniform Rational B-Splines), ovvero curve e superfici parametriche, hanno aperto un universo di forme complesse che, fino alla comparsa delle tecnologie CAD/CAM, erano molto difficili da concepire, sviluppare e rappresentare, figuriamoci fabbricare. Un nuovo universo formale a sua volta ha spinto una ricerca di nuove tettoniche che avrebbero reso costruibili le nuove pelli ondulate e sinuose (con budget ragionevoli)".

6 Ibid.

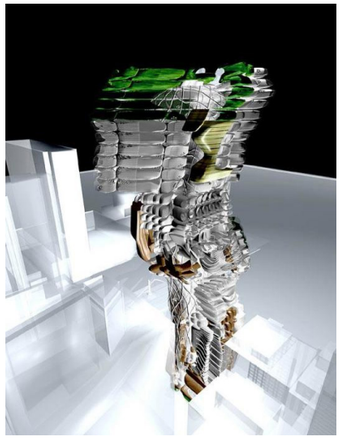


Fig. 1.2_7. Resi-Rise Skyscraper di Kol-Mac Studio, poster della mostra 'Architetture non standard' del 2003, presso il Centre Pompidou (https://www.centrepompidou.fr).

'BLOB' di Greg Lynn[1], del 'FreshH2O eXPO' dei NOX[2] o del 'Salt Water Pavilion' di Kas Oosterhuis[3]. Questi e altri team di architettura[4] parteciparono alla mostra "Architectures non standard", tenutasi presso il Centre Pompidou di Parigi nel 2003-4, che è divenuta il manifesto di queste ricerche sull'applicazione di strumenti digitali alla progettazione architettonica[5].

Il termine non standard che dà il titolo alla mostra fu preso in prestito da Bernard Cache (che aveva partecipato alla mostra come parte del gruppo Objectile, insieme a Patrick Beaucé) secondo cui, grazie alle tecnologie digitali, gli oggetti non sono più progettati ma calcolati, consentendo la progettazione di forme complesse con superfici a curvatura variabile e ponendo le basi per un modo di produzione "non standard"[6]. I suoi objectiles[7] non sono oggetti ma algoritmi, funzioni parametriche che possono determinare un'infinita varietà di oggetti, tutti diversi ma tutti simili, poiché la funzione sottostante è la stessa per tutti[8]: la modifica di parametri di progettazione consente infatti la produzione di forme diverse nella stessa serie, rendendo così possibile la personalizzazione di massa, ovvero la produzione industriale (con macchine a controllo numerico) di oggetti unici[9].

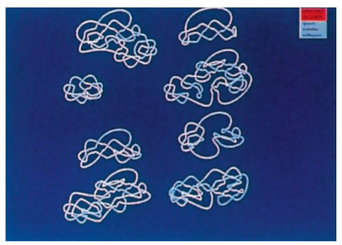


Fig. 1.2_8. Bernard Cache/Objectile, algorithmic knots. (Carpo, M. (2013). The Digital Turn in Architecture 1992-2012).

1 Lynn, G. (1996). "Blobs (or Why Tectonics Is Square and Topology Is Groovy)," ANY 14, 58–62; Lynn, G. (1998). Folds, Bodies & Blobs: collected essays, La lettre volée, Bruxelles.

2 Spuybroek, L. (1998). 'Motor Geometry', Hypersurface Architecture, Stephen Perrella (guest-editor), AD Profile 133, AD 68, 49–55.

3 Oosterhuis, K. (1998). 'Salt Water Live, Behaviour of the Salt Water Pavilion', Stephen Perrella (guest-editor), Hypersurface Architecture, AD Profile 133, AD 68, 56–61.

4 Alla mostra parteciparono: Asymptote, dECOI Architects, DR_D, Greg Lynn FORM, Kol/Mac Studio, Kovac Architecture, NOX, Objectile, Oosterhuis.nl, R&Sie, Servo, UN studio.

5 https://www.centrepompidou.fr/cpv/resource/cpBeKA/rRREoKn

6 Cache, B. (1995). Earth Moves: The Furnishing of Territories. Cambridge: MIT Press, citato in Kolarevic, B. (ed.) Architecture in the digital age: design and manufacturing, 85.

7 Sui concetti di objectile e non standard vedi: Cache, B. (1999). "Objectile: The Pursuit of Philosophy by Other Means?" Architectural Design, vol. 69, no. 9/10, 66-71. Vedi anche il catalogo della mostra "Architectures non standard", tenutasi presso il Centre Pompidou di Parigi nel 2003-4: Migayrou F. (ed) (2004), Architectures non standard, Centre Pompidou, Parigi; vedi inoltre: Perrella, S. (2013). "Bernard Cache/Objectile: Topological Architecture and the Ambiguous Sign - AD May–June 1998", in: Carpo, M. (ed). The Digital Turn in Architecture 1992-2012, John Wiley & Sons.; Carpo, Mario (2011), The Alphabet and the Algorithm, The MIT Press, Cambridge, Mass.-London; Corbellini, G. (2013). "Ultimo tango a Zagarol", in: Corbellini, G., Morassi, C. (ed). Parametrico Nostrano, LetteraVentidue, 109-123.

8 "(…) Deleuze's and Cache's objectile ranks to this day among the most apt definitions of the new technical object in the digital age: the objectile is not an object but an algorithm—a parametric function which may determine an infinite variety of objects, all different (one for each set of parameters) yet all similar (as the underlying function is the same for all)". Carpo, Mario (2011), The Alphabet and the Algorithm, 40.

9 Kolarevic, B. (ed.) Architecture in the digital age: design and manufacturing, 85.

La teoria della piega di Deleuze[1] è stata poi reinterpretata da diversi autori, da Lynn[2] a Eisenman[3], diventando il fondamento teorico di questa stagione di ricerche nell'ambito di quello che quello che viene generalmente definito design parametrico. In queste ricerche Mario Carpo riconosce una doppia eredità: da un lato infatti, la stessa biografia di alcuni dei protagonisti di questa stagione ci porta a riconoscere una continuità con il movimento decostruttivista; dall'altro, la logica del non standard suggerisce anche un'affinità col modello postmoderno di variabilità rizomatica, complessità e frammentazione, in opposizione alla standardizzazione modernista[4].

Architettura e codice

Tuttavia, la vera svolta non fu determinata tanto dai software di modellazione 3d in sé, ma dalla possibilità di estenderne le funzionalità attraverso codici scritti dall'utente stesso. La necessità degli architetti di codificare le loro architetture per sfuggire ai limiti del software tradizionale era stata espressa da Yessios già alla fine degli anni '80, dopo la sua esperienza di progettazione con Eisenman:

"[…] It was becoming apparent that what we were talking about could only be done on the computer. Thus we almost had no choice. The design process had to include the development of new computer tools". "[…] It is probable that the whole design process might have been even more productive and imaginative if the designers were also the developers of the tools they considered desirable" [5].

1 Deleuze, G. (1993). The Fold: Leibniz and the Baroque, trans. Tom Conley, Minneapolis: University of Minnesota Press. Edizione originale francese: Le Pli: Leibniz et le Baroque (Paris: Minuit, 1988).

2 Lynn, G. (2013). "Architectural Curvilinearity: The Folded, the Pliant and the Supple - AD March–April 1993", in: Carpo, M. (ed). The Digital Turn in Architecture 1999-2012, John Wiley & Sons.

3 Eisenman, P. (2013). "Visions Unfolding: Architecture in the Age of Electronic Media - AD September–October 1992", in: Carpo, M. (ed). The Digital Turn in Architecture 1999-2012, John Wiley & Sons.

4 Carpo, M. (2013). "Twenty Years of Digital Design", 10.

5 Chris I. Yessios, citato in: Rocker, I.M. (2008)." Architectures of the Digital Realm: Experimentations by Peter Eisenman | Frank O. Gehry", Die Realität des Imaginären. Architektur und das digitale Bild, 10 Internationales Bauhaus Kolloquium, Weimar, Bauhaus- Universität Weimar, 249-262: 259. Traduzione: "[…] Stava diventando evidente che ciò di cui stavamo parlando poteva essere fatto solo sul computer. Quindi quasi non avevamo scelta. Il processo di progettazione doveva includere lo sviluppo di nuovi strumenti informatici". "[...] È probabile che l'intero processo di progettazione avrebbe potuto essere ancora più produttivo e fantasioso se i progettisti fossero stati anche gli sviluppatori degli strumenti che consideravano desiderabili".

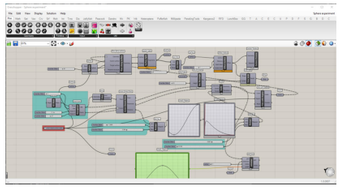


Fig. 1.2_9. Interfaccia di Rhino/Grasshopper. (https://www.grasshopper3d.com/).

Gli sviluppatori di AutoCAD, nel 1986, implementano quindi nel software AutoLISP, 'dialetto' del più famoso linguaggio di programmazione funzionale del 1958 LISP[1]. Nel 1997, Autodesk ha introdotto MaxScript per 3D Studio Max R2 e nel 1998, Alias Systems Corporation ha introdotto il linguaggio di scripting MEL per il software Maya[2]. Rhinoceros, nella versione 3.0 rilasciata nel 2003, incorpora il linguaggio RhinoScript basato su Visual Basic[3]. La possibilità di un lavoro collaborativo e indipendente dalla posizione su Internet ha generato comunità di sviluppo software open source e ha permesso la diffusione di prodotti come Blender (2003) e FreeCAD (2002), oltre che di nuovi linguaggi di programmazione (e relativi dialetti) ai fini della progettazione grafica e spaziale tra cui Processing (2001), OpenFrameworks (2005), Grasshopper (2007), Cinder (2010) e le loro porte web, tra cui Processing.js (2008) e WebGL (2011)[4]. Internet ha non solo consentito un facile accesso a questi strumenti, ma ha anche fornito un mezzo per creare e aggregare in modo collaborativo la loro documentazione tecnica, portando alla nascita di nuova generazione di architetti/programmatori [5]. Il fenomeno dello "scripting", o per meglio dire "tooling", ha permesso dunque che la tecnologia digitale divenisse, da mezzo utilizzato passivamente, un vero strumento progettuale[6].

La diffusione di conoscenze informatiche tra gli architetti e lo sviluppo di linguaggi di programmazione semplificati ha permesso anche di implementare e diffondere ricerche avviate dai pionieri degli anni '60 e '70. Molti dei modelli computazionali utilizzati nella prassi architettonica oggi seguono infatti le tradizioni delle prime ricerche nel campo dell'intelligenza artificiale, come il paradigma di controllo top-down cibernetico o il paradigma bottom-up, basato su regole, dei sistemi formali[7]. Questi paradigmi sono stati ulteriormente sviluppati e ibridati all'interno degli istituti di istruzione e ricerca architettonica avanzata[8] dove, nel frattempo, si sono avviate anche numerose ricerche/tesi di dottorato/tesi di laurea sull'applicazione di tecniche di machine learning e deep learning all'architettura[9].

1 Kalay, Architecture's New Media, 53.
2 Marinčić, Towards Communication In Caad, 68
3 Pugnale, A. (2012). "Engineering Architecture: Come il virtuale si fa reale", Bloom, n.14, 20.
4 Marinčić, Towards Communication In Caad, 68
5 Ibid, 68-69
6 Pugnale, A. (2012). "Engineering Architecture: Come il virtuale si fa reale", Bloom, n.14, 20.
7 Marinčić, Towards Communication In Caad, 69.
8 Ibid.
9 Vedi ad esempio: Chaillou, S. (2020). IA & ARCHITECTURE (Vol. 9). Pavillon de l'Arsenal; Huang, W., & Zheng, H. (2018). "Architectural drawings recognition and generation through machine learning", ACADIA 2018: Recalibration. On imprecisionand infidelity, in: Proceedings of the 38th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), Mexico City, Mexico 18-20 October, 2018, 156-165; Wang, T. C., Liu, M. Y., Zhu, J. Y., Tao, A.,

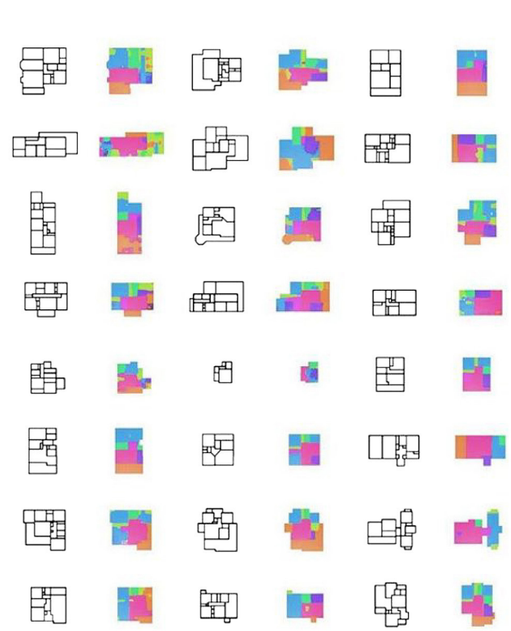


Fig. 1.2_10. Tesi di dottorato di Nathan Peters presso la Harvard Graduate School of Design: utilizza i GAN (generative adversarial network) per progettare la disposizione delle stanze in una casa unifamiliare. (Peters, N. (2018). "Enabling Alternative Architectures: Collaborative Frameworks for Participatory Design").

Kautz, J., & Catanzaro, B. (2018). "High-resolution image synthesis and semantic manipulation with conditional gans", In Proceedings of the IEEE conference on computer vision and pattern recognition, 8798-8807; Peters, N. (2018). Enabling alternative architectures: collaborative frameworks for participatory design (Doctoral dissertation, Harvard University Graduate School of Design); Alonso, N. M. (2017). Suggestive Drawing Among Human and Artificial Intelligences (Doctoral dissertation, Harvard University Graduate School of Design);

1.3_Sistemi computazionali e metodologia progettuale

Design Methods Movement

Come accennato nel paragrafo precedente, le prime ricerche accademiche sui sistemi computazionali per la progettazione architettonica iniziano negli anni '60, intrecciandosi spesso con ricerche sulla metodologia progettuale.
La necessità di un approccio strutturato alla progettazione architettonica e di una razionalizzazione del processo progettuale nasce con la rivoluzione industriale e si afferma con maggiore chiarezza a partire dagli anni '50 e '60[1]. In questi anni infatti l'emergere degli studi su ricerca operativa, processi decisionali e tecniche creative, insieme allo sviluppo della programmazione informatica e dei primi studi sull'intelligenza artificiale, hanno avuto una grande influenza sulle origini dell'interesse sulla metodologia progettuale[2]. Il nuovo "Design Methods Movement" si è sviluppato attraverso una serie di conferenze negli anni '60 e '70[3]. In questo periodo apparvero anche i primi metodi di progettazione o libri di metodologia[4] e i primi libri sulla creatività[5]. Come già accennato, Christopher Alexander, architetto e matematico, nonché uno dei pionieri del movimento, fornisce una base matematica per la sua teoria nel suo libro del 1964 Notes on the Synthesis of Form, in cui descrive l'uso sistematico di un metodo computazionale[1] per mettere insieme ed elaborare le informazioni relative al progetto, strutturandole in modo tale da renderle immediatamente accessibili per la sintesi delle soluzioni progettuali[2].

Metodi di seconda generazione

All'inizio degli anni '70 si registra un periodo di rifiuto della metodologia progettuale, anche da parte di alcuni pionieri del campo tra cui lo stesso Alexander[3], dovuto in parte al clima socioculturale della fine degli anni '60 (le proteste studentesche, il nuovo umanesimo liberale e il rifiuto dei valori precedenti), in parte dovuto anche agli insuccessi delle prime applicazioni di metodi "scientifici" alla progettazione[4]. La metodologia progettuale è stata salvata, tuttavia, dalla proposta di Horst Rittel[5] di considerare quelli degli anni '60 come metodi di "prima generazione" (che naturalmente, con il senno di poi, sembravano un po' semplicistici, ma nondimeno era stato un inizio necessario) svincolando così dal loro esempio le future ricerche che, negli anni '80, trovano un rinnovato interesse, anche grazie agli sviluppi dell'intelligenza artificiale[6].
Molti metodi di progettazione computazionale imitano i metodi tradizionali utilizzati dai progettisti[7]. Tuttavia, la potenza di calcolo e archiviazione del computer permette la generazione e la verifica di un numero elevato di potenziali soluzioni, nonché la raccolta e ricerca all'interno di grandi database di "regole" e precedenti progettuali[8].

Metodi procedurali

I primi metodi computazionali a essere impiegati nella progettazione furono i metodi procedurali[9]. Questi mediano tra l'abilità del progettista di specificare condizioni locali, come le relazioni tra un piccolo numero di variabili, e l'abilità del computer di applicare e testare queste relazioni su set di variabili di numero molto maggiore[10]. Una delle prime questioni affrontate con questi metodi è stata la ge-

1 Kalay, 200-203
2 Cross, N. (2007). "Forty years of design research". Design Studies, 28, pp. 1-4. Vedi anche: Cross N. (1993). "A History Of Design Methodology", M. J. de Vries et aL (eds.), Design Methodology and Relationships with Science, Kluwer Academic Publishers, 15-27.
3 Londra, 1962 (Jones, J. C. and D. G. Thornley, (ed.) (1963), Conference on Design Methods. Oxford, UK, Pergamon Press); Birmingham, 1965 (Gregory, S. A, (ed.) (1966), The Design Method. London, Butterworth Press); Portsmouth, 1967 (Broadbent, G. and A. Ward, (ed.) (1969), Design Methods in Architecture. London, UK, Lund Humphries); Massachusetts, 1969 (Moore, G. T., (ed.) (1970), Emerging Methods in Environmental Design and Planning. Cambridge, Ma., MIT Press); Londra, 1973; New York, 1974 (Spillers, W. R, (ed.) (1974), Basic Questions of Design Theory. Amsterdam/New York, North-Holland/Elsevier); Berkeley, CaL, 1975; Portsmouth, 1976 (Evans, B., J. Powell, R. Talbot, (ed.) (1982), Changing Design. Chichester, UK, John Wiley & Sons Ltd.) e di nuovo nel 1980 (Jacques, R. and J. Powell, (ed.) (1981), Design:Science:Method. Guildford, UK, Westbury House). Cfr. Cross N. (1993). "A History Of Design Methodology".
4 Hall, A D. (1962), A Methodology for Systems Engineering. Princeton, NJ, Van Nostrand; Asimow, M. (1962), Introduction to Design. Englewood Cliffs, NJ., Prentice-Hall; Alexander, C. (1964), Notes on the Synthesis of Form. Cambridge, Ma., Harvard University Press; Archer, L. B. (1965), Systematic Method for Designers. London, The Design Council; Jones, J. C. (1970), Design Methods. Chichester, UK, John Wiley & Sons Ltd; Broadbent, G. (1973), Design in Architecture. Chichester, UK, John Wiley & Sons Ltd. Vedi anche: Rittel, H. (1973), "The State of the Art in Design Methods." Design Research and Methods (Design Methods and Theories) 7(2): 143-147; Rittel, H. and M. Webber (1973), "Dilemmas in a General Theory of Planning." Policy Sciences 4: 155-169; Archer, L. B. (1979), "Whatever Became of Design Methodology?" Design Studies 1(1): 17-18; Cross, N. (1989), Engineering Design Methods. Chichester, UK, John Wiley & Sons Ltd; Roozenburg, N. and N. Cross (1991), "Models of the Design Process: integrating across the disciplines." Design Studies 12(4): 215-220.
5 Gordon, W. J. J. (1961), Synectics. New York, Harper & Row; Osborn, A F. (1963), Applied Imagination - Principles and Procedures of Creative Thinking. New York, Scribener's Sons.

1 Kalay, Architecture's New Media, 66
2 Ibid., 200
3 'I've disassociated myself from the field ... There is so little in what is called "design methods" that has anything useful to say about how to design buildings that I never even read the literature anymore ... I would say forget it, forget the whole thing ... If you call it "It's A Good Idea To Do", I like it very much; if you call it " A Method", I like it but I'm beginning to get turned off; if you call it "A Methodology", I just don't want to talk about it'. Alexander, C (1971). "The state of the art in design methods", DMG Newsletter Vol 5 No 3, 3-7, citato in: Cross, N. (1993). "A History Of Design Methodology", 16.
4 Cross, N. (2007). "Forty years of design research", 2.
5 Rittel,H. (1973). "The state of the art in design methods", Design Research and Methods (Design Methods and Theories) Vol 7 No 2, 143-147.
6 Cross N. (1993). "A History Of Design Methodology", 16-17.
7 Kalay, Architecture's New Media, 237
8 Ibid.
9 Ibid.
10 Ibid.

nerazione automatica di piante architettoniche. Mitchell, Steadman e Liggett[1], nel 1976, propongono un sistema per generare tutte le possibili piante da un dato set di stanze, da cui l'architetto poteva scegliere la più adatta per un determinato progetto[2], utilizzando l'enumerazione completa ("complete enumeration")[3]. Tuttavia, "l'esplosione combinatoria"[4] derivante dall'uso di questo metodo lo rendeva decisamente poco pratico da utilizzare[5]. Cionondimeno il problema, poi definito "space allocation", è rimasto uno dei più controversi, interessanti e difficili della progettazione computazionale e sono state proposte molte possibili soluzioni[6].

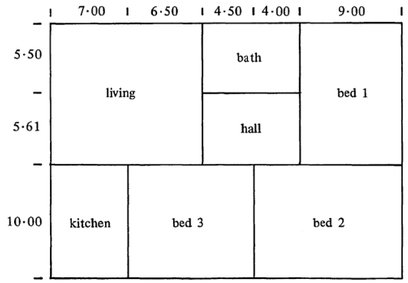


Figura 1.3_1 Esempio di generazione automatica di piante nello studio di Mitchell, Steadman e Liggett. (Fonte: Mitchell, W. J., Steadman, J. P., & Liggett, R. S. (1976). "Synthesis and optimization of small rectangular floor plans".)

In termini generali, il problema consiste nella generazione del layout degli spazi e delle attività che ospitano all'interno di un edificio in base a un qualche criterio

1 Mitchell, W. J., Steadman, J. P., & Liggett, R. S. (1976). "Synthesis and optimization of small rectangular floor plans". Environment and Planning B: Planning and Design, 3(1), 37-70.

2 Kalay, Architecture's New Media, 241. Vedi anche: Gero, J. S. (1977). "Note on 'Synthesis and Optimization of Small Rectangular Floor Plans' of Mitchell, Steadman, and Liggett". Environment and Planning B: Planning and Design, 4(1), 81-88.

3 "Complete enumeration (…) exploit the ability of the computer to generate a vast number of solutions to a problem – potentially all possible solutions – to be examined by the designer (or another computational procedure) one at the time for their fit with the function and the context (or any other condition, for that matter)". Kalay, 238.

4 "In mathematics, a combinatorial explosion is the rapid growth of the complexity of a problem due to how the combinatorics of the problem is affected by the input, constraints, and bounds of the problem. Combinatorial explosion is sometimes used to justify the intractability of certain problems". Da Wikipedia: https://en.wikipedia.org/wiki/Combinatorial_explosion

5 Kalay, Architecture's New Media, 241

6 Vedi ad esempio: Armour, G. C., & Buffa, E. S. (1963). "A heuristic algorithm and simulation approach to relative location of facilities". Management Science, 9(2), 294-309; Whitehead, B. and Eldars, M. Z. (1964). "An Approach to the Optimum Layout of Single-Story Buildings.". Architects' Journal, 17, 1373–1379; Lee, R. C. and Moore, J. M. (1967). "CORELAP Computerized Relationship Layout Planning.". Journal of industrial Engineering, vol. 18(3), 195–200; Agraa, O. M., & Whitehead, B. (1968). Nuisance restrictions in the planning of single-storey layouts. Building Science, 2(4), 291-302; Shaviv, E., & Gali, D. (1974). "A model for space allocation in complex buildings: an aproach". Building Industrial 6, 493-518.

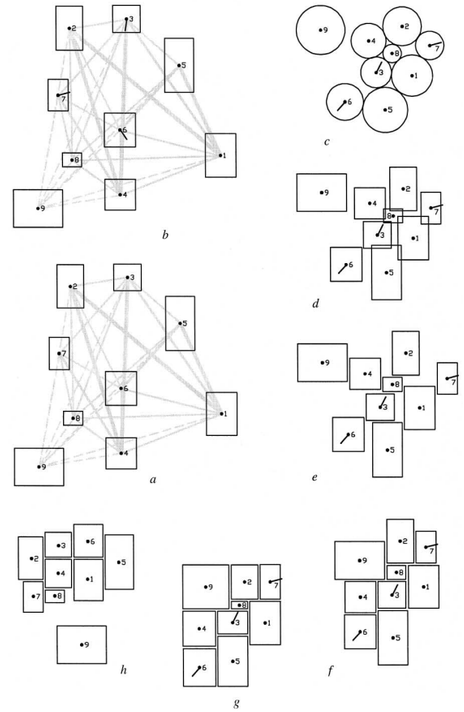


Figura 1.3_2 Esempio di space allocation nel lavoro di Arvin e House.
(Fonte: Arvin, S. A., & House, D. H. (2002). "Modeling architectural design objectives in physically based space planning").

razionale (solitamente, la minimizzazione delle distanze tra spazi che ospitano funzioni collegate)[7]. Sebbene le ricerche sulla space allocation siano state sviluppate principalmente tra gli anni '60 e '80[8], alcune ricerche sono ancora in corso in questo campo (un esempio è il lavoro di Michalek et al.[9]).

Metodi euristici

I metodi procedurali hanno un approccio sistematico che richiede una completa comprensione del lavoro da svolgere. Questo è difficile da avere in architettura, dove i metodi adottati di solito si affidano alle esperienze personali e professionali

7 Kalay, Architecture's New Media, 241-247

8 Una rassegna sull'uso degli algoritmi di space allocation è stato fatto da Frew in: Frew, S.R. (1980). "A survey of space allocation algorithms in use in architectural designin the past twenty years". In: Proceedings of the 17th Conference on DesignAutomation. Minneapolis, Minnesota, 165-174

9 Michalek, J, Choudhary, R. Papalambros, P. (2002). "Architectural layout design optimization". Engineering Optimization 34(5), 461-484.

accumulate dai progettisti.[1] La formalizzazione di un tale metodo "soft", che affronta problemi "mal strutturati"[2] e attinge a conoscenze approssimate piuttosto che precise, è stata l'area di interesse dei metodi euristici del tipo formalizzato dalla ricerca sull'intelligenza artificiale[3]: diversamente dai metodi procedurali, i metodi euristici non possono garantire che arriveranno alla soluzione del problema a cui sono applicati, né che questa sarà ottimale[4]. Nonostante ciò, o forse a causa, i metodi euristici sono capaci di risolvere problemi che i metodi procedurali non possono, in quanto fanno affidamento su una visione globale, olistica del problema anziché su una visione "localizzata"[5].

Knowledge-based systems

L'IA ha ispirato la ricerca, nell'ambito della progettazione architettonica, di sistemi basati sulla conoscenza ("knowledge-based") capaci di sintetizzare soluzioni progettuali sulla base di analogie, casi studio, regole progettuali derivate dalle conoscenze di progettisti esperti, e perfino trasformazioni formali capaci di generare soluzioni all'interno di un riconosciuto corpo di lavoro architettonico[6].

In alcuni casi i metodi euristici utilizzati lavorano per analogia con altri campi disciplinari[7]. Nei lavori di Steadman[8] e Schwartz[9], ad esempio, si utilizza la metafora elettrica per guidare il processo di sintesi progettuale, mentre Arvin e House[10] usano una metafora meccanica[11].

Tuttavia, i metodi knowledge-based tendono generalmente a replicare i metodi tradizionali più utilizzati dagli architetti per guidare il processo progettuale: l'uso di riferimenti progettuali e sistemi di regole compositive[12].

1 Kalay, Architecture's New Media, 255

2 Simon, H.A. (1973). "The structure of ill structured problems", Artificial Intelligence 4,181–201.

3 Kalay, Architecture's New Media. Vedi anche: Eastman, C. M. (1969). "Cognitive processes and ill-defined problems: A case study from design", Proceedings of the International Joint Conference on Artificial Intelligence: IJCAI, Vol. 69, 669-690.

4 Kalay, Architecture's New Media, 255-256.

5 Ibid., 256.

6 Ibid., 255-256.

7 Ibid., 256-261.

8 March, L., & Steadman, P. (1974). The geometry of environment: an introduction to spatial organization in design, MIT Press, 263-264.

9 Schwarz, A., Berry, D. M., & Shaviv, E. (1994). Representing and solving the automated building design problem. Computer-aided design, 26(9), 689-698.

10 Arvin, S. A., & House, D. H. (2002). "Modeling architectural design objectives in physically based space planning". Automation in Construction, 11(2), 213-225.

11 Kalay, Architecture's New Media, 256-261.

12 Kalay, Architecture's New Media.

Un approccio molto popolare, soprattutto negli anni '90, è quello basato sul case-based reasoning (CBR)[1], un processo di risoluzione di nuovi problemi che si basa su soluzioni trovate in passato per problemi simili: in questo caso le decisioni progettuali sono guidate dal confronto tra il problema e un caso precedente[2].

Il processo dunque consiste nell'individuare il caso più appropriato, determinare i parametri che devono essere modificati per adattarlo al problema dato e infine sviluppare l'algoritmo per eseguire queste modifiche[3]. Una delle maggiori difficoltà nell'uso di questo metodo è la costruzione della libreria di casi a cui fare riferimento e la relativa indicizzazione, necessaria alla selezione del caso da utilizzare[4]. Esempi di sistemi case-based sono SEED[5] (Software Environment to Support Early Phases in Building Design) di Flemming, Electronic Cocktail Napkin[6] di Gross e Do e il formalismo ICF[7] (Issue-Concept-Form) di Oxman[8].

Tuttavia, secondo Heylighen e Neuckermans[9], non sono stati compiuti progressi convincenti e nessuno degli approcci sviluppati sembra aver influenzato profondamente la pratica architettonica[10]. In termini di generazione formale, Grobman et al.[11] identificano un altro tipo di approccio basato sull'idea che sia possibile definire una sorta di template parametrico di un edificio che può essere utilizzato per generare diversi edifici simili, riconoscendo un esempio di tale strumento in Variomatics di Oosterhuis[12].

Case-based reasoning

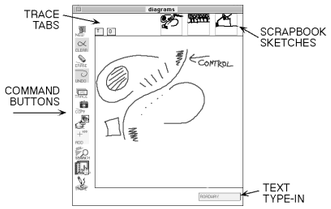


Figura 1.3_3 Interfaccia dell' Electronic Cocktail Napkin di Gross
(Fonte: Gross, M. D. (1996). "The electronic cocktail napkin- a computational environment for working with design diagrams")

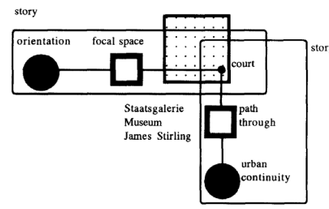


Figura 1.3_4 Formailismo Issue-Concept-Form di Oxman per la Staatsgalerie di J. Stirling
(Fonte: Oxman, R. E. (1994). "Precedents in design: a computational model for the organization of precedent knowledge")

1 Riesbeck, C. K., & Schank, R. C. (1989). Inside Case-Based Reasoning, Lawrence Erlbaum Associates. Hillsdale, NJ, US.

2 Grobman, Y.J., Yezioro, A. & Capeluto I.G. (2009). "Computer-Based Form Generation in Architectural Design - a Critical Review", International Journal of Architectural Computing issue 04, volume 07, 535-553: 540; Kalay, Architecture's New Media, 261-267.

3 Grobman et al. "Computer-Based Form Generation in Architectural Design - a Critical Review", 540;

4 Kalay, Architecture's New Media, 261-267.

5 Flemming, U. (1994). "". Automation in construction, 3(2-3), 123-133.

6 Gross, M. D. (1996). "The electronic cocktail napkin- a computational environment for working with design diagrams". Design studies, 17(1), 53-69.

7 Oxman, R. E. (1994). "Precedents in design: a computational model for the organization of precedent knowledge". Design studies, 15(2), 141-157.

8 Kalay, Architecture's New Media, 261-267.

9 Heylighen,A. & Neuckermans, H. (2001). "A case base of Case-Based Design tools for architecture". Computer-Aided Design, 33(14), 1111-1122.

10 Grobman et al. "Computer-Based Form Generation in Architectural Design - a Critical Review", 540.

11 Ibid.

12 Oosterhuis, K. (2002). Programmable Architecture. L'Arcaedizioni.

Sistemi esperti

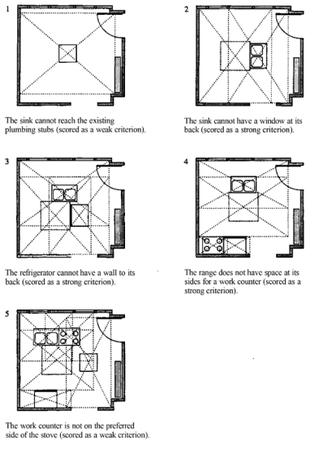


Figura 1.3_5 Esempi di regole di valutazione utilizzate dal tester di LOOS
(Fonte: Flemming et al. (1988). "A generative expert system for the design of building layouts – Version 2")

Piuttosto che cercare di catturare le conoscenze e l'esperienza dei progettisti attraverso i loro lavori in forma di casi, è possibile catturarle direttamente sottoforma di regole progettuali[1]. Questo metodo è da sempre parte della cultura architettonica e sistemi di regole per la progettazione ci sono stati tramandati da Vitruvio, Alberti, Vignola e Palladio, tra gli altri. La sua implementazione computazionale è avvenuta attraverso i sistemi esperti[2]. In questi sistemi le conoscenze di professionisti esperti vengono codificate in costrutti if-then che descrivono le azioni da intraprendere quando si incontrano certe condizioni[3].
Esempi di sistema esperto basato su regole sono LOOS, sviluppato da Flemming et al.[4], che comprende un generatore di possibili piante architettoniche e un tester che le valuta in base a un set di regole, estendibile dall'utente, contenute nel sistema[5]; e PREDIKT (PREliminary Design of KITchens), di Oxman[6], un sistema esperto per la progettazione di cucine che permette sia la valutazione che la generazione delle piante.

L'uso di conoscenze euristiche permette ai sistemi esperti una maggiore flessibilità rispetto ai programmi procedurali ma, al contrario di questi, non possono garantire il raggiungimento di una soluzione per un dato problema[7]. Tuttavia, se la soluzione esiste, i sistemi esperti sono in grado di trovarla molto più velocemente in quanto il loro ragionamento euristico può sfruttare delle "scorciatoie" per arrivare alla soluzione, piuttosto che dover esaminare tutte le possibili soluzioni in modo esaustivo[8]. Le shape grammars funzionano in maniera analoga ai sistemi esperti, con la differenza che sostituiscono sia la condizione che l'azione dell'if-then rule con costrutti geometrici[1]. Come anticipato nel primo paragrafo di questo capitolo, a tale approccio tradizionale di tipo "esperto", che è stato oggetto di numerose ricerche e implementazioni che verranno analizzate nei prossimi capitoli (2,3 e 4), si è col tempo affiancato un approccio alternativo che prevede l'uso di grammatiche "ingenue" in combinazione con tecniche di apprendimento automatico (machine learning), che verrà illustrato nel capitolo 5.

La possibilità di incorporare processi di apprendimento nei sistemi knowledge-based è stata vista infatti come la migliore risposta alle difficoltà inerenti l'utilizzo di tali sistemi già illustrate nei paragrafi precedenti[2]. Coyne e Postmus[3] hanno individuato tre vantaggi fondamentali che potrebbero derivare da questo tipo di approccio:

"It is important to model learning in knowledge-based systems intended for design applications for several reasons. First, automated learning may obviate the considerable human effort required to code knowledge into computers. Second, a system with the ability to learn could be expected to improve its performance. This improvement would be evident the next time the system was called upon to perform a similar task. Third, it can be argued that learning is a crucial part of the design process. Design appears to be a process of exploration and discovery. This suggests that the act of designing may also involve the definition of the space of possibilities through which that search takes place"[4].

1 Kalay, Architecture's New Media, 267.
2 Kalay, Architecture's New Media, 267.
3 Ibid., 267-270. Vedi anche: Hayes-Roth, F. (1985). "Rule-based systems". Communications of the ACM, 28(9), 921-932; Hayes-Roth, F. (1984). "Knowledge-based expert systems". Computer, (10), 263-273. Un approfondimento su questo argomento è dato nel capitolo 2, paragrafo 2.2 (Sistemi esperti e sistemi di produzione)
4 Flemming, U., Coyne, R., Glavin, T., & Rychener, M. (1988). "A generative expert system for the design of building layouts – Version 2". In: Gero, J.S. (ed) Artificial Intelligence in Engineering: Design, Amsterdam: Elsevier. Vedi anche: Flemming, U. (1994). "Artificial intelligence and design: a mid-term review". In: Carrara, G., & Kalay, Y. E. (Eds.). Knowledge-based computer-aided architectural design. Elsevier Science Inc.
5 Kalay, Architecture's New Media, 270.
6 Oxman, R. E. (1992). "Multiple Operative And Interactive Modes In Knowledge-Based Design Systems". In: Kalay, Y.E. (ed) Principles of Computer-Aided Design: Evaluating and Predicting Design Performance, John Wiley&Sons, 125-143.
7 Kalay, Architecture's New Media, 268.
8 Ibid.

1 Ibid., 201-202; Flemming, Ulrich (1987). "The Role of Shape Grammars in the Analysis and Creation of Designs", in: Y. Kalay (ed.), *The Computability of Design*, New York, Wiley Interscience, 245–272.
2 Cfr.: Rafiq, M. Y., Bugmann, G., & Easterbrook, D. J. (2001). "Neural network design for engineering applications". Computers & Structures, 79(17), 1541-1552.
3 Coyne, R. D., & Postmus, A. G. (1990). "Spatial applications of neural networks in computer-aided design", Artificial intelligence in Engineering, 5(1), 9-22.
4 Ibid., 9. Traduzione "È importante modellare l'apprendimento nei sistemi basati sulla conoscenza destinati ad applicazioni di progettazione per diversi motivi. Primo, l'apprendimento automatizzato può ovviare al considerevole sforzo umano richiesto per codificare la conoscenza nei computer. In secondo luogo, ci si può aspettare che un sistema con la capacità di apprendere migliori le sue prestazioni. Questo miglioramento sarà evidente la prossima volta che il sistema sarà chiamato a svolgere un compito simile. In terzo luogo, si può sostenere che l'apprendimento è una parte cruciale del processo di progettazione. La progettazione sembra essere un processo di esplorazione e scoperta. Ciò suggerisce che l'atto di progettare può anche coinvolgere la definizione dello spazio di possibilità attraverso il quale avviene quella ricerca".

Secondo la definizione di Samuel[1], l'apprendimento automatico è un'applicazione dell'intelligenza artificiale che fornisce ai sistemi la capacità di apprendere e migliorare automaticamente dall'esperienza, senza essere programmati esplicitamente. Tale capacità implica, evidentemente, un'autonomia del computer che da strumento metodologico può diventare un vero e proprio agente creativo nel processo progettuale. Come vedremo nel prossimo paragrafo, tuttavia, il contributo che i sistemi computazionali possono dare a un processo creativo può essere declinato in molti modi e non vi è un punto di vista unanime sull'argomento.

1 Samuel, Arthur L. (1959). "Some Studies in Machine Learning Using the Game of Checkers". IBM Journal of Research and Development. 44: 206–226.

1.4_Creatività e computer

I metodi discussi nel paragrafo precedente hanno lo scopo di aiutare gli architetti a razionalizzare il processo progettuale e, generalmente, non sono pensati per produrre soluzioni "creative", ma piuttosto per guidare attività progettuali ordinarie[1]. La maggior parte dei metodi progettuali sono infatti destinati a produrre soluzioni adeguate, non eccezionali[2].
Ciononostante, molti sforzi sono stati e vengono ancora fatti per stabilire come i metodi computazionali di progettazione possano contribuire al processo creativo e se possano, eventualmente, *essere* creativi.
La risposta a queste domande dipende, naturalmente, da cosa si intende per creatività.

Pensiero creativo

Il pensiero creativo è stato definito come la capacità di rompere le modalità di pensiero convenzionali, l'"inerzia psicologica"[3] che intralcia il pensiero comune, e di utilizzare quello che deBono[4] ha definito il "pensiero laterale"[5].
Questa concezione della creatività come facoltà umana che supera i processi cognitivi ordinari ha le sue radici nel romanticismo ed è stata al centro dell'interesse delle scienze cognitive, l'insieme delle discipline (intelligenza artificiale, psicologia, linguistica, filosofia, neuroscienze e antropologia) che hanno per oggetto lo studio dei processi cognitivi umani e artificiali[6]. Secondo Coyne[7] i primi modelli computazionali di tali processi sviluppati in quest'area di ricerca risultavano poco interessanti proprio perché mancava l'ingrediente cognitivo più importante: la creatività.

"I am taking cognitive science as the study that emerged from artificial intelligence and reductive psychology, taking computation as a

1 Kalay, Architecture's New Media, 223
2 Ibid., 224
3 Altshuller, G. S. (1984). Creativity as an exact science: the theory of the solution of inventive problems. Gordon and Breach.
4 De Bono, E. (1967). The use of lateral thinking. Penguin.
5 Coyne, R D ; Rosenman, M. A.; Radford, A D ; and Gero, J. S. (1987). "Innovation and Creativity in Knowledge-Based CAD", in J. S. Gero (ed.), Expert Systems in Cornputer- Aided Design, Amsterdam: North-Holland, 435-465.
6 Enciclopedia Treccani, "Scienza cognitiva": http://www.treccani.it/enciclopedia/scienza-cognitiva (visitato il 09/08/2020). Vedi anche: Miller, G.A. (2003). "The cognitive revolution: A historical perspective", Trends in Cognitive Sciences. 7 (3), 141-144.
7 Coyne, R. (1997). "Creativity as commonplace", Design Studies, 18(2), 135-141. Vedi anche: Felgenbaum, E.A. & Feldman, J. (eds) (1995). Computers and thought, MIT Press, Cambridge, MA.

paradigm of reason. (…) Influenced strongly by logical positivism, some of its exponents say that whether or not the mind is knowable is a metaphysical question and beyond resolution - let us simply make computational models and test them against some aspect of human behaviour. (…) The results of its experiments were so mundane that it decided there must be a higher cognitive function, ingredient X, that turns the simple block stacking, theorem proving, decision tree searching, exercises into something interesting"[1].

Per integrare questa proprietà nei sistemi computazionali è stato necessario identificarne in modo più preciso le caratteristiche, cosa la rende possibile e in che modo opera. Innanzi tutto, è stato suggerito, l'esito di un processo creativo dovrebbe portare a un risultato nuovo e di valore, ma anche inaspettato[2].
Ma come si arriva a un risultato di questo tipo? Come è strutturato il processo?
Alcuni dei tentativi di comprendere e automatizzare la creatività di maggiore successo sono fondati sulle nozioni di esplorazione e trasformazione di uno spazio di possibili soluzioni[3]. La studiosa di intelligenza artificiale Margaret Boden individua infatti tre tipi fondamentali di creatività - combinatoria, esplorativa e trasformativa - riconoscendo negli ultimi due quelli più adatti alla modellazione computazionale.

"Combinational creativity produces unfamiliar combinations of familiar ideas, and it works by making associations between ideas that were previously only indirectly linked (…) Exploratory creativity rests on some culturally accepted style of thinking, or "conceptual space" (…) In exploratory creativity, the person moves through the space, exploring it to find out what's there (including previously unvisited locations)— and, in the most interesting cases, to discover both the potential and the limits of the space in question. These are the "most interesting" cases because they may lead on to the third form of creativity, which can be the most surprising of all. In transformational creativity, the space or style itself is transformed by altering (or dropping) one or more of its defining dimensions. As a result, ideas can now be generated that simply could not have been generated before the change"[1].

I processi creativi combinatori sono sostanzialmente processi linguistici, ma non riguardano solo il linguaggio verbale. Possono infatti manifestarsi attraverso l'associazione di elementi di qualunque insieme omogeneo (lettere dell'alfabeto, parole, forme geometriche, etc.). Tuttavia, contrariamente a quanto si potrebbe pensare, questo è il tipo di creatività più difficile da modellare per l'IA perché, per quanto il computer sia capace di creare nuove combinazioni di concetti familiari, più difficilmente riesce a generare o selezionare tali combinazioni in modo tale che la maggior parte siano interessanti[2]. Risultati migliori si ottengono con i processi esplorativi in cui il computer ha dimostrato di saper raggiungere risultati paragonabili a quelli di professionisti umani altamente competenti[3].
All'interno di questo paradigma della creatività computazionale, un sistema può trovare soluzioni nuove o sorprendenti, attraverso la ricerca all'interno di uno spazio

1 Ibid, 138. Traduzione: "Prendo la scienza cognitiva come studio emerso dall'intelligenza artificiale e dalla psicologia riduttiva, prendendo il calcolo come paradigma della ragione. (…) Influenzati fortemente dal positivismo logico, alcuni dei suoi esponenti affermano che se la mente è conoscibile o meno è una domanda metafisica e al di là della risoluzione - cerchiamo semplicemente di creare modelli computazionali e testarli contro alcuni aspetti del comportamento umano. (…) I risultati dei suoi esperimenti furono così banali che decise che doveva esserci una funzione cognitiva superiore, l'ingrediente X, che trasforma il semplice impilamento di blocchi, la dimostrazione del teorema, la ricerca dell'albero decisionale, gli esercizi in qualcosa di interessante".

2 Gero, J. S. (1996). "Creativity, emergence and evolution in design". Knowledge-Based Systems, 9(7), 435-448: 435. Vedi anche: Boden, M. (1991). The Creative Mind, Myths and Mechanisms, Wiedenfeld and Nicholson, London; Gero J.S. and Maher, M.L. (1992). "Mutation and analogy to support creativity in computer-aided design", in G.N. Schmitt (ed.), CAAD Futures '91, Vieweg, Wiesbaden, 261-270; Gero J.S. and Maher, M.L. (eds.) (1993). Modeling Creativity and Knowledge-Based Creative Design, Lawrence Erlbaum, Hillsdale, N J; Kim, S.H. (1990). Essence of Creativity, Oxford University Press, New York; Sternberg, R. (ed.) (1988) The Nature of Creativity, Cambridge University Press, Cambridge; Weisberg, R.W. (1986). Creativity: Genius and Other Myths, W.H. Freeman, New York.

3 Kelly, N., & Gero, J. S. (2015). Situated interpretation in computational creativity. Knowledge-Based Systems, 80, 48-57; Boden, M. (1991). The Creative Mind, Myths and Mechanisms, Wiedenfeld and Nicholson, London; Langley, P., Simon, H. A., Bradshaw, G. L., & Zytkow, J. M. (1987). Scientific discovery: Computational explorations of the creative processes. MIT press.

1 Boden, M.A. (2009). "Computer Models of Creativity", AI Magazine Vol 30, No 3, 23-34: 24-25. Traduzione: "La creatività combinatoria produce combinazioni non familiari di idee familiari e funziona creando associazioni tra idee che in precedenza erano solo indirettamente collegate (...) La creatività esplorativa si basa su uno stile di pensiero culturalmente accettato, o "spazio concettuale" (...) Nella creatività esplorativa, la persona si muove attraverso lo spazio, esplorandolo per scoprire cosa c'è lì (compresi i luoghi precedentemente non visitati) - e, nei casi più interessanti, per scoprire sia il potenziale che i limiti dello spazio in questione. Questi sono i casi "più interessanti" perché possono portare alla terza forma di creatività, che può essere la più sorprendente di tutte. Nella creatività trasformazionale, lo spazio o lo stile stesso viene trasformato alterando (o rilasciando) una o più delle sue dimensioni definitive. Di conseguenza, ora è possibile generare idee che semplicemente non avrebbero potuto essere generate prima del cambiamento".

2 Ibid., 25.

3 Ibid., 27.

definito o attraverso l'esplorazione che trasforma questo spazio in qualche modo[1].

Design creativo

Si è provato in vari modi di incorporare queste idee nei metodi di progettazione computazionali. Nell'ambito della ricerca sui sistemi knowledge-based per l'architettura, Coyne et al.[2] hanno proposto una distinzione tra il concetto di innovazione e quello di creatività. L'innovazione riguarda l'esplorazione di uno spazio già definito da un corpo di conoscenze, mentre la creatività si manifesta quando questo spazio è solo parzialmente definito.

"In the case of innovation the space being explored can be infinite in extent, such as that defined by a linguistic grammar. The generation of a sentence in a language can be regarded as innovative. In the case of creativity we are concerned with the search for the 'paraphernalia' that defines that space. In the case of a natural language (considering only its syntax) we could say that the 'paraphernalia' defining the space of all possibile discourses is its grammar. To produce a discourse which also involved the production of a new grammar would be regarded as creative. (…) We generalize the idea of creativity in design by contending that the 'paraphernalia' which defines search spaces in design is knowledge. As well as the search for designs, creativity therefore involves the search for the knowledge defining the space within which that search should take place. This view appears to capture something of the richness, the complexity and even the intractibility of design as a process"[3].

1 Kelly, N., & Gero, J. S. (2015). Situated interpretation in computational creativity. Knowledge-Based Systems, 80, 48-57; Wiggins, G. A. (2006). Searching for computational creativity. New Generation Computing, 24(3), 209-222; Gero, J.S. (1994). "Towards a model of exploration in computer-aided design", in: J.S. Gero, E. Tyugu (eds.), Formal Design Methods for CAD, North-Holland, Amsterdam, 315–336; Boden, M. (1991). The Creative Mind, Myths and Mechanisms, Wiedenfeld and Nicholson, London.

2 Coyne, R.D. ; Rosenman, M. A.; Radford, A D ; and Gero, J. S. (1987). "Innovation and Creativity in Knowledge-Based CAD", in J. S. Gero (ed.), Expert Systems in Cornputer- Aided Design, Amsterdam: North-Holland, 435-465.

3 Coyne et al. (1987). "Innovation and Creativity in Knowledge-Based CAD", 461-462.Traduzione: "Nel caso dell'innovazione lo spazio esplorato può essere di estensione infinita, come quello definito da una grammatica linguistica. La generazione di una frase in una lingua può essere considerata innovativa. Nel caso della creatività, siamo interessati alla ricerca dell"armamentario' che definisce quello spazio. Nel caso di un linguaggio naturale (considerando solo la sua sintassi) potremmo dire che l"armamentario' che definisce lo spazio di tutti i discorsi possibili è la sua grammatica. Produrre un discorso che implichi anche la produzione di una nuova grammatica sarebbe considerato creativo. (…) Generalizziamo l'idea di creatività nel design sostenendo che l"armamentario' che definisce gli spazi di ricerca nella progettazione è la conoscenza. Così come la ricerca delle soluzioni progettuali, la creatività implica quindi la ricerca della conoscenza che definisce lo spazio entro il quale quella ricerca dovrebbe svolgersi. Questa visione sembra catturare qualcosa della ricchezza, della complessità e persino dell'intrattabilità della progettazione come processo".

La progettazione creativa è pertanto un processo in cui deve essere scoperta anche la conoscenza che definisce lo spazio di esplorazione. Ciò implica la capacità del sistema di acquisire conoscenza, di controllare i propri processi e modificare la propria struttura[1]. Quindi, di trasformare lo spazio di esplorazione.
Gero descrive un concetto analogo quando parla di progettazione creativa come di una "perturbazione dello schema":

"(…) we can describe routine designing as following a defined schema where the expectations of what follows is defined by the schema. Creative designing, which is part of non-routine designing, can be described as perturbing the scheme to produce unexpected and incongruous results"[2].

La proprietà che permette questa "perturbazione", è l'emergenza:

"A property that is only implicit, i.e. not represented explicitly, is said to be an emergent property if it can be made explicit. Emergence is considered to play an important role in the introduction of new schemas and consequently new variables. Emergence is a recognised phenomenon in visual representations of structure. It maps directly onto the concept of changing schemas since a new schema is generally needed to describe the emergent property"[3].

1 Ibid.

2 Gero, J. S. (1996). "Creativity, emergence and evolution in design". Knowledge-Based Systems, 9(7), 435-448: 435. Traduzione: "(…) possiamo descrivere la progettazione di routine come una che segue uno schema definito in cui le aspettative di ciò che segue sono definite dallo schema stesso. La progettazione creativa, che fa parte della progettazione non di routine, può essere descritta come una perturbazione dello schema per produrre risultati inaspettati e incongrui".

3 Gero, J. S. (1996). "Creativity, emergence and evolution in design", 438. Traduzione: "Una proprietà che è solo implicita, cioè non rappresentata in modo esplicito, si dice che sia una proprietà emergente se può essere resa esplicita. Si ritiene che l'emergenza svolga un ruolo importante nell'introduzione di nuovi schemi e di conseguenza nuove variabili. L'emergenza è un fenomeno riconosciuto nelle rappresentazioni visive della struttura. Si associa direttamente al concetto di modifica degli schemi poiché un nuovo schema è generalmente necessario per descrivere la proprietà emergente".

Emergenza

Nella progettazione architettonica l'emergenza è un fenomeno noto: nel pensare per schizzi, disegni, diagrammi, la specifica soluzione progettuale non viene fissata all'inizio, ma emerge mentre si svolge il processo[1]. Tale emergenza di forma e significato da una rappresentazione altrimenti non-strutturata (o differentemente strutturata) è l'essenza del design creativo[2]. Essa offre al progettista un feedback visivo e un meccanismo per riconoscere e comprendere le possibilità e le implicazioni del progetto in evoluzione, quindi una base per il suo successivo sviluppo ed evoluzione in una soluzione progettuale completa[3].

Il concetto di emergenza ha le sue radici nella filosofia del XIX secolo, in particolare nell'opera di John Stuart Mill[4] e George Henry Lewes[5] e nel British Emergentism, ed è diventato popolare a partire dagli anni '90, soprattutto grazie al lavoro di Holland[6]. In termini molto generali, il termine viene usato per indicare la creazione di qualcosa le cui proprietà non sono esplicitamente rappresentate o identificate in ciò che l'ha generato[7]. Ciò appare evidente nei sistemi complessi: il "comportamento globale" di questi sistemi infatti emerge a partire dalle interazioni tra i singoli elementi che li compongono, in un modo non prevedibile dallo studio degli elementi stessi[8]. Questo esempio ci aiuta a individuare due implicazioni fondamentali del concetto di emergenza: la prima risiede in quei caratteri di novità e imprevedibilità che abbiamo visto essere tipici dei processi creativi:

"Every genuine emergent introduces novelty into the world"[9].

1 Kalay, Architecture's New Media, 281; vedi anche: Habraken, N.J. (1985). The Appearance of the Form, four essays on the position designing takes between people and things, Cambridge, Awater Press.

2 Kalay, Architecture's New Media, 281.

3 Kalay, Architecture's New Media, 281; Schon, D. A. (1983). The Reflective Practitioner: How Professionals Think in Action. Basic Books: N. Y.; vedi anche: Oxman, R. (2002). "The thinking eye: visual re-cognition in design emergence". Design Studies, 23(2), 135-164.

4 Mill, J. S. (1843). System of Logic, Longmans, Green, Reader, and Dyer, London.

5 Lewes, G. H. (1875). Problems of Life and Mind. Osgood and Company, Boston, MA.

6 Holland, J. H. (1998). Emergence: from chaos to order. Perseus, Cambridge, MA.

7 Knight, T. (2003). "Computing with emergence". Environment and Planning B: planning and design, 30(1), 125-155: 125-126.

8 "Emergence means complex systems do things that you would not predict or expect given a knowledge of the separate agents that make them up. The system has characteristics that none of the agents do. The whole, as the old saying goes, is greater than the sum of its parts". Agar, M. (1999). "Complexity theory: An exploration and overview based on John Holland's work". Field Methods, 11(2), 99-120: 106.

9 Goudge, T. A. (1972). "Emergent evolutionism", The Encyclopedia of Philosophy, Macmillan, New York, 474-477: 475. Traduzione: "Ogni vero emergente introduce novità nel mondo"

"[An emergent property is] unpredictable not only on the basis of the knowledge available prior to its emergence but even on the basis of ideally complete knowledge of the state of the cosmos prior to its emergence" [1].

La seconda è l'idea che vi sia quella che Knight definisce una "gerarchia di livelli" alla base del fenomeno dell'emergenza:

"Emergence comes about through the combination of fixed primitives at one level to form an emergent entity, with new properties, at the next higher level"[2].

Potremmo dire quindi che è l'esito di processo bottom-up, "dal basso verso l'alto". Questa idea era già presente nelle teorie dei filosofi emergentisti[3] ed è stata poi ripresa anche da Holland, il cui modello computazionale di emergenza si basa sulla formalizzazione delle macro-leggi che descrivono il comportamento dei fenomeni emergenti derivanti da livelli inferiori[4].

"these new [macro] laws apply to complex phenomena that are consequences of the original laws; they are at a new level. We will gain a deeper understanding of emergence if we can deepen our understanding of this concept of levels of definition"[5].

Emergenza nei sistemi computazionali

Secondo Holland lo studio dell'emergenza è strettamente legato alla capacità di specificare un dominio ampio e complesso tramite un piccolo insieme di leggi o

1 Ibid. Traduzione: "[una proprietà emergente è] imprevedibile non solo sulla base della conoscenza disponibile prima della sua apparizione, ma anche sulla base di una conoscenza idealmente completa dello stato del cosmo prima della sua apparizione".

2 Knight, T. (2003). "Computing with emergence". 129.

3 "The higher quality emerges from the lower level of existence and has its roots therein, but it emerges therefrom, and it does not belong to that lower level, but constitutes its possessor a new order of existent with its special laws of behavior". Alexander, S. (1920). *Space, Time, and Deity*, Macmillan, London, 46.

4 Knight, T. (2003). "Computing with emergence", 126.

5 Holland, J. H. (1998). Emergence: from chaos to order, 190, citato in: Knight, T. (2003). "Computing with emergence". Traduzione: "queste nuove [macro] leggi si applicano a fenomeni complessi che sono conseguenze delle leggi originali; sono a un nuovo livello. Acquisiremo una comprensione più profonda dell'emergenza se possiamo approfondire la nostra comprensione di questo concetto di livelli di definizione".

regole[1] e questa capacità è oggi fondamentale per molti sistemi computazionali[2]. In architettura, i sistemi basati su regole, dagli automi cellulari a shape grammar, capaci di supportare l'emergenza, si sono rivelati strumenti di progettazione molto potenti[3].

Per shape grammar infatti l'emergenza è una caratteristica fondamentale[4] ma in un modo che differisce da altri modelli computazionali. Come questa si manifesta è una questione che verrà affrontata nello specifico nel secondo capitolo, tuttavia è utile anticipare alcuni aspetti.

Alcune differenze risiedono nell'uso e nel riconoscimento di questa proprietà[5]. In particolare, gioca un ruolo chiave la natura non deterministica di shape grammar, già anticipata nel primo paragrafo, che permette di riconoscere e operare su forme che non sono predefinite ma piuttosto emergono in qualunque momento e da una qualunque parte delle forme generate attraverso l'applicazione delle regole[6]. Pertanto, contrariamente a quanto accade in altri sistemi computazionali, l'emergenza in shape grammar non è "gerarchica"[7]. Inoltre, come vedremo nel capitolo 2, in shape grammar l'emergenza può variare da completamente prevedibile a completamente imprevedibile.

Emergenza e imprevedibilità

In effetti, l'equiparazione che si riscontra in molta letteratura tra i concetti di emergenza e imprevedibilità è stata messa in discussione da alcuni autori. Secondo Knight infatti, benché l'emergenza apparentemente spontanea di fenomeni non esplicitamente integrati nelle regole sia spesso equiparata all'imprevedibilità e alla sorpresa, queste non sono essenziali perché vi sia emergenza nei sistemi computazionali[8].

Kalay si spinge oltre e arriva ad affermare che per quanto sia i metodi procedurali come space allocation che euristici come il case-based reasoning e i sistemi esperti possano produrre soluzioni creative, nel senso di *inaspettate*, per via del gran numero di soluzioni che sono in grado di generare, tuttavia tali soluzioni sono pur sempre

1 Holland, J. H. (1998). Emergence: from chaos to order, 123.

2 Knight, T. (2003). "Computing with emergence", Environment and Planning B: planning and design, 30(1), 125-155.

3 Knight, T. (2003). "Computing with emergence". 130

4 Knight, T. (2003). "Computing with emergence"; Stiny, G. (1994). "Shape rules: closure, continuity, and emergence", Environment and Planning B 21, S49-S78; March, L. (1996). "Babbage's miraculous computation revisited", Environment and Planning B: Planning and Design 23, 369- 376; March, L. (1996). "Rulebound unruliness", Environment and Planning B: Planning and Design 23, 391-399.

5 Knight, T. (2003). "Computing with emergence",127.

6 Kalay, Architecture's New Media, 274.

7 Knight, T. (2003). "Computing with emergence", 129.

8 Ibid., 132

prevedibili, in quanto sono conosciuti e prevedibili i loro processi[1].

Per qualificare queste soluzioni come veramente creative e inaspettate, dobbiamo rivolgerci a metodi computazionali capaci di generare soluzioni innovative senza cercare di imitare la creatività umana o incorporare conoscenze pre-elaborate[2]. Pertanto, sorgono due problemi quando veniamo a formalizzare il processo di emergenza in un metodo computazionale: come generare soluzioni nuove e "inaspettate" e come riconoscere tra queste le più significative o "interessanti" una volta che sono emerse senza far ricorso a modelli "umani"[3]. Una possibilità risiede, come abbiamo visto, nell'apprendimento automatico, che permette una maggiore autonomia del computer:

"Learning is often viewed as the most fundamental aspect of intelligence, as it enables the agent to become independent of its creator. It is an essential component of an agent design whenever the designer has incomplete knowledge of the task environment. Therefore, learning provides autonomy in that the agent is not dependent on the designer's knowledge for its success and can free itself from the assumptions built into its initial configuration. Learning may also be the only route by which we can construct very complex intelligent systems. In many application domains, the best systems are constructed by a learning process rather than by traditional programming or knowledge engineering"[4].

Inevitabilmente, le diverse opinioni riguardo all'idea di *imprevedibilità* dipendono dalle diverse sfumature di significato che si possono dare a questo concetto. La possibilità di una creatività *autonoma* del computer è ovviamente particolarmente interessante. Tuttavia, se ragioniamo in termini di contributo del computer ai processi

1 Kalay, *Architecture's New Media*, 279-280.

2 Ibid., 280.

3 Ibid., 281.

4 Russell, S. (1996). "Machine Learning", in M. Boden (ed.), ***Artificial Intelligence***, Academic Press. Traduzione: "L'apprendimento è spesso visto come l'aspetto più fondamentale dell'intelligenza, poiché consente all'agente di diventare indipendente dal suo creatore. È una componente essenziale della progettazione di un agente ogni volta che il progettista ha una conoscenza incompleta dell'ambiente del compito. Pertanto, l'apprendimento fornisce autonomia in quanto l'agente non dipende dalla conoscenza del progettista per il suo successo e può liberarsi dai presupposti incorporati nella sua configurazione iniziale. L'apprendimento può anche essere l'unica via attraverso la quale possiamo costruire sistemi intelligenti molto complessi. In molti domini applicativi, i migliori sistemi sono costruiti da un processo di apprendimento piuttosto che dalla programmazione tradizionale o dall'ingegneria della conoscenza".

creativi, i metodi computazionali hanno dimostrato di permettere di raggiungere soluzioni non (o difficilmente) raggiungibili con metodi tradizionali, anche quando si lavora con modelli di creatività che imitano quella umana.
Ciò su cui sembra esserci una sorta di accordo unanime, comunque, è la relazione tra emergenza e creatività, motivo per cui ci interessa particolarmente la capacità di shape grammar, riconosciuta dallo stesso Kalay[1], di permettere l'emergenza.

1 Ibid., 274.

CAPITOLO 2

SHAPE GRAMMAR

In questo capitolo viene discussa shape grammar come sistema di produzione basato su regole. Vengono dunque prima sottolineate analogie e differenze con la grammatica generativa chomskiana e i L-systems a cui si ispira (paragrafo 2.1). Nel paragrafo 2.2 viene spiegato il funzionamento di shape grammar, come si costruiscono le regole e viene spiegata la differenza tra shape grammar "tradizionale" e la versione parametrica. Nel paragrafo 2.3 viene illustrato come si manifesta l'emergenza in shape grammar.

2.1_ Sistemi di riscrittura

Nell'articolo "Shape Grammars and the Generative Specification of Painting and Sculpture"[1] pubblicato nel 1971, Stiny e Gips presentano un metodo per la generazione di forme che ha come componente strutturale primaria shape grammar, un sistema di produzione[2] che si ispira alla grammatica formale di Chomsky[3].
Infatti, come nella grammatica generativa, il processo generativo di shape grammar si basa sulla riscrittura, una tecnica che serve a definire oggetti complessi sostituendo successivamente parti di un oggetto iniziale utilizzando un insieme di regole o produzioni[4]. Tuttavia, mentre il sistema di riscrittura utilizzato da Chomsky per descrivere le caratteristiche sintattiche dei linguaggi naturali opera su stringhe di caratteri, shape grammar opera su forme geometriche:

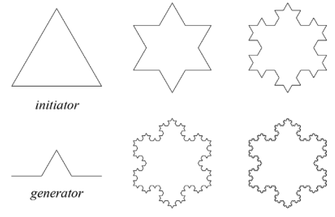


Figura 2.1_1 Un classico esempio di oggetto grafico definito in termini di regole di riscrittura è la curva del fiocco di neve, proposta nel 1906 da H. von Koch.
(Koch, H. (1906). "Une méthode géométrique élémentaire pour l'étude de certaines questions de la théorie des courbes planes", Acta mathematica, 30, 145-174.)

"Shape grammars are similar to phrase structure grammars, which were introduced by Chomsky in linguistics. Where phrase structure grammars are defined over an alphabet of symbols and generate one-dimensional strings of symbols, shape grammars are defined over an alphabet of shapes and generate n-dimensional shapes. The definition of shape grammars follows the standard definition of phrase structure grammars"[5].

Tale approccio grammaticale alla generazione di forme geometriche era stato anticipato dal lavoro di Aristid Lindenmayer, un biologo che aveva introdotto un nuovo tipo di meccanismo di riscrittura delle stringhe, successivamente denominato Lindenmayer systems – o L-systems – finalizzato alla modellazione delle piante[6]. In origine, lo studio si concentrava principalmente sulla topologia delle piante, i loro aspetti geometrici andavano oltre lo scopo della teoria. Successivamente, sono state

1 Stiny, G. & Gips, J. (1972). "Shape grammars and the generative specification of painting and sculpture", in Petrocelli, O.R. (ed.) The Best Computer Papers of 1971, Auerbach, Philadelphia, 125-135.
2 Cfr. Capitolo 1, paragrafo 1.1.
3 Chomsky, N. (1957). Syntactic structures,'s-Gravenhage: Mouton. The Hague, The Netherlands.
4 Prusinkiewicz, P. & Lindenmayer, A. (1991). *The Algorithmic Beauty of Plants,* Springer Verlag, 2.
5 "Le shape grammars sono simili alle grammatiche della struttura delle frasi, introdotte da Chomsky in linguistica. Laddove le grammatiche della struttura delle frasi sono definite su un alfabeto di simboli e generano stringhe unidimensionali di simboli, le shape grammars sono definite su un alfabeto di forme e generano forme n-dimensionali. La definizione delle shape grammars segue la definizione standard delle grammatiche della struttura delle frasi".
6 Lindenmayer, A. (1968). "Mathematical models for cellular interaction in development, Parts I and II", Journal of Theoretical Biology, 18, 280–315.

Figura 2.1_2 Esempio di pianta generata da un L-system
(Fonte: Prusinkiewicz, P. & Lindenmayer, A. (1991). The Algorithmic Beauty of Plants)

n=5, δ=18°

ω : **plant**
p_1 : **plant** → **internode** + [**plant** + **flower**] − − //
[− − **leaf**] **internode** [+ + **leaf**] −
[**plant flower**] + + **plant flower**
p_2 : **internode** → F **seg** [// & & **leaf**] [// ∧ ∧ **leaf**] F **seg**
p_3 : **seg** → **seg** F **seg**
p_4 : **leaf** → [' { +f−ff−f+ | +f−ff−f }]
p_5 : **flower** → [& & & **pedicel** ' / **wedge** //// **wedge** ////
wedge //// **wedge** //// **wedge**]
p_6 : **pedicel** → FF
p_7 : **wedge** → [' ∧ F] [{ & & & & −f+f | −f+f }]

proposte diverse interpretazioni geometriche dei L-systems al fine di trasformarli in uno strumento versatile per la modellazione[1].
La differenza essenziale tra grammatiche di Chomsky e L-systems risiede nel metodo di applicazione delle produzioni: nelle grammatiche di Chomsky le produzioni sono applicate in sequenza, una alla volta, mentre nei L-systems sono applicate in parallelo, cioè sostituiscono simultaneamente tutte le lettere in una data parola[2]. Questa differenza riflette la motivazione biologica di questi ultimi: le produzioni hanno lo scopo di catturare le divisioni cellulari negli organismi multicellulari, dove possono verificarsi molte divisioni allo stesso tempo[3]. L'applicazione di produzione parallela ha un impatto essenziale sulle proprietà formali dei sistemi di riscrittura. Ad esempio, ci sono linguaggi che possono essere generati da sistemi L context-free (chiamati sistemi OL) ma non da grammatiche di Chomsky context-free[4].

1 Prusinkiewicz, P. & Lindenmayer, A. (1991). The Algorithmic Beauty of Plants, 1.
2 Ibid., 2-3.
3 Ibid.
4 Ibid.; Herman, G.T. & Rozenberg, G. (1975). Developmental systems and languages. North-Holland, Amsterdam; Salomaa, A. (1973). Formal languages. Academic Press, New York.

In shape grammar le regole di produzione possono essere applicate sia in serie (come con le grammatiche di Chomsky) che in parallelo, in modo simile al modo in cui le produzioni sono fatte nei L-Systems. Del resto lo stesso Stiny precisa che il riferimento alla grammatica di Chomsky è volutamente non letterale:

"My idea was that a grammar had a limited number of rules that could generate an unlimited number of different things, and that the resulting language was the set of things the rules produced. I didn't think that designs were sentences, but instead that grammars could generate both sentences and designs—whatever they were"[1].

Stiny sottolinea infatti che le grammatiche di Chomsky non potrebbero essere efficacemente applicate ad oggetti con due o più dimensioni:

"[Chomsky hierarchy] provides a very useful taxonomy for generative grammars, but it doesn't work for shapes made up of lines, etc. Perhaps this is the fatal flaw in visual calculating—'Wouldn't it be wonderful if we could classify shape grammars the way Chomsky classifies generative grammars.' Complexity is measured by counting symbols (units), and this is fine for shapes containing points and other things that are zero dimensional, for example, sets with members. They have definite descriptions that keep them numerically distinct. Nonetheless, there are no units for higher-dimensional things. The shapes in rules don't have parts that I can count before I calculate. These depend on how the rules are used. The description that makes sense for a rule now isn't binding later. What the rule is doing may change freely as it's applied"[2].

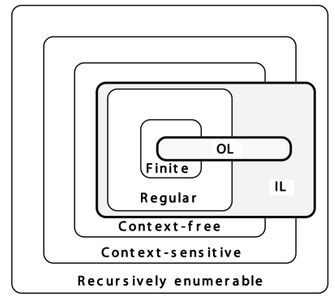


Figura 2.1_3 Grafico che illustra la relazione tra grammatica di Chomsky e L-system
(Fonte: Prusinkiewicz, P. & Lindenmayer, A. (1991). The Algorithmic Beauty of Plants)

1 Stiny, G. (2006). *Shape: talking about seeing and doing*, MIT Press, 18. Traduzione: "La mia idea era che una grammatica avesse un numero limitato di regole che potevano generare un numero illimitato di cose diverse e che il linguaggio risultante fosse l'insieme di cose che le regole producevano. Non pensavo che i design fossero frasi, ma invece che la grammatica potesse generare sia frasi che design, qualunque cosa fossero".
2 Ibid., 271. Traduzione: "[La gerarchia di Chomsky] fornisce una tassonomia molto utile per grammatiche generative, ma non funziona per forme composte da linee, ecc. Forse questo è il difetto fatale nel calcolo visivo: "Non sarebbe meraviglioso se potessimo classificare la forma grammatiche nel modo in cui Chomsky classifica le grammatiche generative '. La complessità viene misurata contando i simboli (unità), e questo va bene per le forme contenenti punti e altre cose che sono a dimensione zero, ad esempio, insiemi con membri. Hanno descrizioni definite che li mantengono numericamente distinti. Tuttavia, non ci sono unità per cose di dimensioni superiori. Le forme nelle regole non hanno parti che posso contare prima di calcolare. Questi dipendono da come vengono utilizzate le regole. La descrizione che ha senso per una regola ora non è vincolante in seguito. Ciò che la regola sta facendo può cambiare liberamente quando viene applicata".

2.2_Shape grammar: definizione formale

Il formalismo di shape grammar viene descritto nel dettaglio da Stiny per la prima volta nel 1980 nell'articolo "Introduction to shape and shape grammars"[1]. Di seguito ne vengono riportate le caratteristiche principali, ponendo l'attenzione in particolare sulla differenza tra shape grammar tradizionale e shape grammar parametrica. Prima di procedere con la descrizione del formalismo di shape grammar, verranno riportate alcune definizioni fondamentali.

Definizioni

Forma

Una forma s è una disposizione finita di linee di lunghezza limitata ma diversa da zero in due o tre dimensioni. Le forme vengono specificate disegnandole in un sistema di coordinate cartesiane.
Questo sistema di coordinate di solito non è dato esplicitamente: la sua origine, gli assi e le unità sono sottintesi. La forma che non contiene linee è chiamata forma vuota ed è indicata da s_φ.

Sottoforma

Una forma s_1 è una sottoforma di una forma s_2 (indicata da $s_1 \leq s_2$) se ogni parte di s_1 fa anche parte di s_2. Cioè, s_1 coincide punto per punto con qualche parte di s_2 nel sistema di coordinate in cui sono disegnate.

Operazioni booleane

L'unione delle forme s_1 e s_2 (indicata con $s_1 + s_2$) è la forma costituita dalle linee sia in s_1 che in s_2. L'unione delle forme delle forme viene definita sovrapponendo i loro disegni in modo che i sistemi di coordinate in cui vengono disegnati coincidano. Le forme s_1 e s_2 sono entrambe sottoforme della forma $s_1 + s_2$.
La differenza delle forme s_1 e s_2 (indicata da $s_1 - s_2$) è la forma ottenuta da s_1 cancellando quella parte di s_1 che coincide con s_2. La forma $s_1 - s_2$ è sempre una sottoforma della forma s_1 ma non è necessario che sia una sottoforma della forma s_2.
L'intersezione delle forme s_1 e s_2 (indicata da $s_1 \bullet s_2$) è la forma costituita solo da quelle linee presenti sia in s_1 che in s_2. La forma $s_1 \bullet s_2$ è una sottoforma della forma s_1 e una sottoforma della forma s_2.

1 Stiny G, 1980 "Introduction to shape and shape grammars", Environment and Planning B, 7, 343-351. I materiali di questo paragrafo sono tratti prevalentemente da questo testo.

Trasformazioni

Le trasformazioni euclidee sono traslazione, rotazione, ridimensionamento e riflessione, o combinazioni finite di queste. Una trasformazione τ di una forma s è la forma indicata da τ (s).

Insiemi di forme

Un insieme finito di forme può essere utilizzato come vocabolario per la formazione di altre forme.
Dato un insieme finito di forme S, l'insieme di forme S^+ è l'insieme minimo contenente le forme in S e che è chiuso sotto l'unione delle forme e le trasformazioni euclidee.
L'insieme delle forme S* è dato da $S^* = S+ \cup \{ s_\varphi \}$.

Forme etichettate

Un punto etichettato p: A è un punto p a cui è associato un simbolo A.
Due punti etichettati p_1: A_1 e p_2: A_2 sono equivalenti se e solo se $p_1 = p_2$ e A_1 è identico a A_2.
Una trasformazione τ di un punto etichettato p: A è il punto etichettato τ (p): A, dove τ (p) è il punto ottenuto applicando τ a p.
Una trasformazione τ di un insieme di punti etichettati P è l'insieme di punti etichettati τ (P) prodotti applicando la trasformazione τ a ciascun punto etichettato in P.
Una forma etichettata è composta da due parti: una forma e un insieme di punti etichettati. Più precisamente, una forma etichettata σ_1 è data da una coppia ordinata $\sigma_1 = (s, P)$, dove s è una forma e P è un insieme finito di punti etichettati.
I punti etichettati in P possono coincidere con le linee in s, ma non è necessario che ciò avvenga.
Una forma etichettata $\sigma = (s, P)$ viene specificata disegnando s in un sistema di coordinate cartesiane e posizionando le etichette in P accanto ai punti a cui sono associate.
Una forma s è la forma etichettata (s, Φ).
Le relazioni e le operazioni sulle forme possono essere estese alle forme etichettate:
Per le forme etichettate $\sigma_1 = (s_1, P_1)$ e $\sigma_2 = (s_2, P_2)$, σ_1 è una sottoforma di σ_2 (indicata da $\sigma_1 \subseteq \sigma_2$) se e solo se $s_1 \subseteq s_2$ e $P_1 \subseteq P_2$.
L'unione di forma di σ_1 e σ_2 (indicata da $\sigma_1 + \sigma_2$) è la forma etichettata $(s_1 + s_2, P_1 + P_2)$.
La differenza di forma di σ_1 e σ_2 (indicata da $\sigma_1 - \sigma_2$) è la forma etichettata $(s_1 - s_2, P_1 - P_2)$.
L'intersezione di forma di σ_1 e σ_2 (indicata con $\sigma_1 \bullet \sigma_2$) è la forma etichettata $(s_1 \bullet s_2, P_1 \bullet P_2)$.

Una trasformazione di forma τ di una forma etichettata $\sigma = (s, P)$ è la forma etichettata $\tau(\sigma)$ data da
$\tau(\sigma) = \tau(\tau(s), \tau(P))$.

Insiemi di forme etichettate $(S, L)^+$ e $(S, L)^*$

Le forme etichettate possono essere formate da determinati vocabolari di forme e simboli. Una forma etichettata σ è composta da forme in un insieme S e simboli in un insieme L ogni volta che ha una delle seguenti tre forme:

1. $\sigma = (s, \Phi)$, dove s è una forma nell'insieme S^+;
2. $\sigma = (s_\varphi, P)$, dove P è un insieme finito, non vuoto di punti etichettati in cui qualsiasi simbolo associato a un punto è un elemento dell'insieme L;
3. $\sigma = (s, P)$, dove s è una forma e P è un insieme di punti etichettati che soddisfano le condizioni (1) e (2), rispettivamente.

L'insieme di tutte le forme etichettate composto da forme nell'insieme S e simboli nell'insieme L è indicato da $(S, L)^+$. L'insieme di forme etichettate $(S, L)^*$ contiene oltre a tutte le forme etichettate nel set $(S, L)^+$ la forma etichettata vuota (s, Φ).

Famiglie di forme

Una famiglia di forme è definita da una forma parametrizzata s, che si ottiene consentendo alle coordinate dei punti finali delle linee massime in una data forma di essere variabili. Un particolare membro di questa famiglia è determinato dall'assegnazione g di valori reali a queste variabili. Questi valori possono essere richiesti per soddisfare determinate condizioni specificate. Il risultato dell'applicazione dell'assegnazione g alla forma parametrizzata s è la forma indicata con g(s).
Le forme parametrizzate e le loro assegnazioni possono essere considerate come generalizzazioni di trasformazioni applicate alle forme. Le assegnazioni a forme parametrizzate possono, in generale, variare qualsiasi aspetto spaziale di una forma, purché le linee rimangano diritte.
Una forma parametrizzata etichettata σ è data da $\sigma = (s, P)$, dove s è una forma parametrizzata e P è un insieme di punti parametrizzati etichettati. La forma etichettata prodotta applicando g a σ è data da $g(\sigma) = (g(s), g(P))$.

Il formalismo di shape grammar

Il formalismo di shape grammar consente di definire gli algoritmi direttamente in termini di forme etichettate e forme etichettate parametrizzate. Ciascuno di questi algoritmi definisce un linguaggio di forme.

Shape grammar

Formalmente, shape grammar è una quadrupla (S, L, R, I), dove:

1. S è un insieme finito di forme.
2. L è un insieme finito di simboli.
3. R è un insieme finito di shape rules (regole di forma) del tipo $\alpha \rightarrow \beta$, dove α è una forma etichettata in $(S, L)^+$ e β è una forma etichettata in $(S, L)^*$.
4. I è una forma etichettata in $(S, L)^+$ chiamata forma iniziale.

Una forma viene generata iniziando con la forma iniziale I e applicando in modo ricorsivo le regole di forma nell'insieme R.
Una regola di forma $\alpha \rightarrow \beta$ si applica a una forma etichettata γ quando vi è una trasformazione τ tale che $\tau(\alpha)$ è una sottoforma di γ, cioè: $\tau(\alpha) \leq \gamma$
La forma etichettata prodotta applicando la regola $\alpha \rightarrow \beta$ alla forma etichettata γ sotto la trasformazione τ è data da $[\gamma - \tau(\alpha)] + \tau(\beta)$. Questa forma etichettata è formata sostituendo l'occorrenza di $\tau(\alpha)$ in γ con $\tau(\beta)$.

Le forme etichettate vengono generate da shape grammar applicando le regole una alla volta alla forma iniziale o alle forme etichettate prodotte da precedenti applicazioni di regole di forma. Una data forma etichettata γ viene generata dalla grammatica se esiste una serie finita di forme etichettate che iniziano con la forma iniziale e terminano con γ in modo tale che ogni termine della serie tranne il primo sia prodotto applicando una regola di forma al suo immediato predecessore.

Una shape grammar definisce un insieme di forme chiamato linguaggio. Questo linguaggio contiene tutte le forme generate dalla grammatica a cui non sono associati simboli, ovvero forme etichettate della forma (s, Φ). Ciascuna di queste forme è derivata dalla forma iniziale applicando le regole di forma; ciascuna è composta da forme o sottoforme nell'insieme S.

Esempio di shape grammar

Nel suo articolo Stiny illustra il funzionamento di shape grammar con un semplice esempio. Si parte da una forma iniziale e due regole di forma. Sia le forme etichettate nelle due regole che la forma iniziale sono costituite da un quadrato e dal simbolo •. La forma etichettata sul lato sinistro di entrambe le regole di forma è costituita da un quadrato e dal simbolo • situato nel punto medio di uno dei suoi bordi. La forma etichettata sul lato destro della prima regola è costituita da questo quadrato e da un altro inscritto in esso. Ogni vertice del quadrato interno coincide con il punto medio di un diverso bordo del quadrato esterno. Il simbolo • si trova nel punto medio di un bordo del quadrato interno. La forma etichettata sul lato destro della seconda regola della forma è costituita dal quadrato sul lato sinistro. La forma iniziale è un quadrato etichettato con il simbolo • al centro di uno dei suoi bordi.

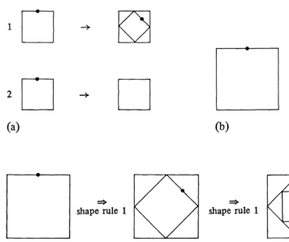


Figura 2.2_1 Esempio di una semplice shape grammat che inscrive quadrati in quadrati, a) regole di forma, b) forma iniziale. (Fonte: Stiny G, 1980 "Introduction to shape and shape grammars")

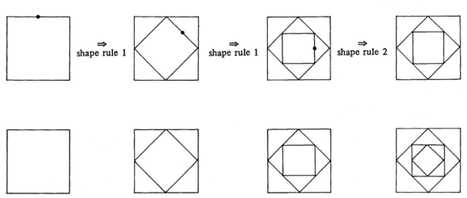


Figura 2.2_2 Generazione di una forma utilizzando la grammatica della forma di figura 2.2_1. (Fonte: Stiny G, 1980 "Introduction to shape and shape grammars")

La prima regola viene applicata alla forma iniziale nel passaggio 1 e alla forma etichettata risultante nel passaggio 2. Questa regola si applica solo alle forme etichettate che contengono un quadrato con il simbolo • associato al punto medio di uno dei suoi bordi. La regola di forma inscrive un quadrato, anch'esso etichettato in questo modo, nel quadrato etichettato corrispondente al suo lato sinistro e cancella il simbolo • ad esso associato. Di conseguenza, il simbolo • è sempre associato al punto medio di uno spigolo del quadrato inscritto più di recente. Pertanto, la prima regola può essere applicata alla forma iniziale e a ciascuna forma etichettata prodotta durante il processo di generazione della forma al massimo una volta. Poiché la forma etichettata sul lato sinistro della prima regola di forma è una sottoforma della forma etichettata sul lato destro, può essere applicata di nuovo a qualsiasi forma etichettata già prodotta.

La seconda regola di forma viene applicata nel passaggio 3 della generazione. Il lato sinistro di questa regola di forma è identico al lato sinistro della prima regola di forma e quindi si applica in circostanze identiche. La seconda regola di forma cancella il simbolo • associato al punto medio di un bordo di un quadrato per produrre una forma nel linguaggio definito dalla grammatica della forma. Gli altri passaggi nella generazione non contengono forme in questo linguaggio, poiché il simbolo • è associato a ciascuna delle forme in questi passaggi. Nessuna regola di forma può essere applicata dopo che è stata utilizzata la seconda regola di forma, poiché entrambe le regole di forma richiedono l'applicazione del simbolo • per essere applicate.

Parametric shape grammar

Le shape grammar parametriche sono un'estensione delle grammatiche di forma in cui le regole di forma vengono definite inserendo termini aperti in uno schema generale. Uno schema di regole di forma $\alpha \rightarrow \beta$ consiste di forme etichettate parametrizzate α e β, dove nessun membro della famiglia di forme etichettate specificate da α è la forma etichettata vuota. Ogni volta che valori specifici vengono attribuiti a tutte le variabili in α e β da un assegnamento g per determinare forme etichettate specifiche, viene definita una nuova regola di forma $g(\alpha) \rightarrow g(\beta)$. Questa regola di forma può quindi essere utilizzata per trasformare una data forma etichettata in una nuova nel solito modo. Più precisamente, se c'è una trasformazione τ che rende $g(\alpha)$ una sottoforma della forma etichettata data, allora questa occorrenza della forma etichettata $\tau [g(\alpha)]$ può essere sostituita con la forma etichettata $\tau [g(\beta)]$. A meno che non sia esplicitamente limitata, τ è una trasformazione generale. Si dirà che uno schema di regole di forma si applica a una forma etichettata ogni volta che definisce una regola di forma che si applica alla forma etichettata.

Esempio di shape grammar parametrica

Le implicazioni di queste generalizzazioni sono illustrate da Stiny nel seguente semplice esempio in cui viene generalizzata la shape grammar illustrata nell'esempio precedente. In questo caso la grammatica invece lavorare con quadrati, lavora con quadrilateri generici. La shape grammar parametrica illustrata in figura 2.2_3 contiene una forma iniziale costituita dal punto etichettato (0, 0): ▪ e tre schemi di regole di forma:
Il primo schema definisce regole di forma che sostituiscono un punto etichettato indicato dal simbolo ▪ con un quadrilatero convesso avente uno dei suoi bordi etichettato dal simbolo •. Il lato sinistro dello schema è costituito da un punto parametrizzato etichettato (x_1, y_1): ▪ ; il lato destro è costituito da un quadrilatero parametrizzato q con vertici (x_1, y_1), (x_2, y_2), (x_3, y_3) e (x_4, y_4) e un punto parametriz-

zato etichettato (x_5, y_5): •.
I valori assegnati alle variabili nello schema soddisfano due condizioni:
a) I punti (x_1, y_1), (x_2, y_2) , (x_3, y_3) e (x_4, y_4) sono i vertici di un quadrilatero convesso.
b) Il punto (x_5, y_5) è il punto medio della linea con estremi (x_3, y_3) e (x_4, y_4).

Il secondo schema definisce le regole di forma che iscrivono un quadrilatero convesso in un altro in modo tale che ciascuno dei vertici di quello interno coincidano con uno spigolo diverso di quello esterno.
Il lato sinistro dello schema è costituito dal quadrilatero parametrizzato q e dal punto parametrizzato etichettato (x_5, y_5): • ;
il lato destro è costituito dal quadrilatero parametrizzato q, un altro quadrilatero parametrizzato r con vertici (x_6, y_6), (x_7, y_7) , (x_8, y_8) e (x_9, y_9) e un punto parametrizzato etichettato (x_{10}, y_{10}): •.
I valori assegnati a queste nuove variabili soddisfano queste condizioni:
c) I punti (x_6, y_6), (x_7, y_7) , (x_8, y_8) e (x_9, y_9) sono coincidenti con le linee aventi punti finali (x_1, y_1) e (x_2, y_2), (x_2, y_2) e (x_3, y_3), (x_3, y_3) e (x_4, y_4) e (x_4, y_4) e (x_1, y_1), rispettivamente, ma non coincidono con questi punti.
d) Il punto (x_{10}, y_{10}) è il punto medio della linea con i punti finali (x_7, y_7) , (x_8, y_8).
Le condizioni (a) e (c) garantiscono che due quadrilateri qualsiasi determinati dall'assegnazione di valori alle variabili in q e r siano entrambi convessi.

Il terzo schema definisce regole di forma che cancellano il simbolo • dal bordo di un quadrilatero convesso. Il lato sinistro dello schema è costituito dal quadrilatero parametrizzato q e dal punto parametrizzato etichettato (x_5, y_5): •; il lato destro è costituito dal quadrilatero parametrizzato q.

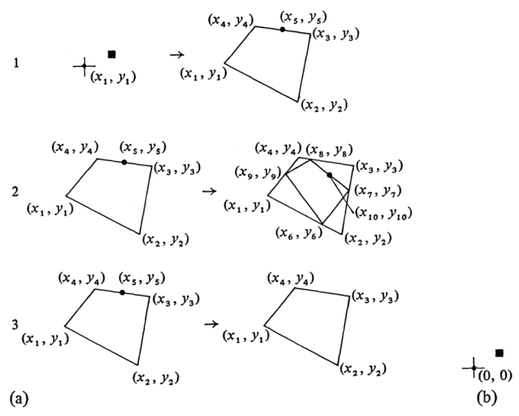


Figura 2.2_3 Esempio di shape grammar parametrica: (a) schemi delle regole di forma, (b) forma iniziale.
(Fonte: Stiny G, 1980 "Introduction to shape and shape grammars")

Il primo schema viene applicato alla forma iniziale nel passaggio 1. Questa regola trasforma la forma iniziale in un quadrilatero convesso etichettato. Il secondo schema viene applicato nei passaggi 2 e 3 alle forme etichettate prodotte dai passaggi 1 e 2. Ciascuna di queste regole è formata da un lato sinistro con un quadrilatero con un bordo contrassegnato dal simbolo • e un lato destro in cui inscrive un nuovo quadrilatero con un bordo contrassegnato dal simbolo • all'interno del primo e cancella il simbolo • ad esso associato. Ciò impedisce l'uso dello schema per iscrivere più di un quadrilatero ad ogni passaggio. Il terzo schema viene applicato nel passaggio 4 alla forma etichettata prodotta nel passaggio 3. Questo schema termina il processo di generazione cancellando il simbolo •.

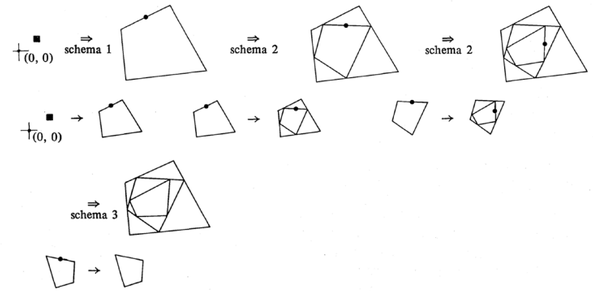


Figura 2.2_5 La generazione di una forma utilizzando la grammatica illustratat in figura 2.2_4
(Fonte: Stiny G, 1980 "Introduction to shape and shape grammars")

Alcune considerazioni

Alla luce dei due esempi, è quindi possibile cogliere le diverse potenzialità dei due tipi di grammatiche (parametrica e non). Mentre una shape grammar tradizionale infatti permette di definire linguaggi formali con relazioni proporzionali determinate da serie aritmetiche o geometriche, i tipi di trasformazioni più generali consentiti nelle shape grammar parametriche conservano le linee rette, ma possono variare le dimensioni relative e gli angoli compresi. Le grammatiche parametriche possono quindi essere utilizzate per definire linguaggi con relazioni proporzionali determinate in qualsiasi modo.

2.3_Emergenza in shape grammar

L'emergenza in shape grammar di solito si riferisce all'identificazione e all'uso, nella computazione, di forme non predefinite nella grammatica, ma che nascono dalle forme generate dalle applicazioni delle regole[1]. Nell'articolo "Computing with emergence", pubblicato nel 2003, Terry Knight illustra il concetto di emergenza in shape grammar attraverso un semplice esempio:

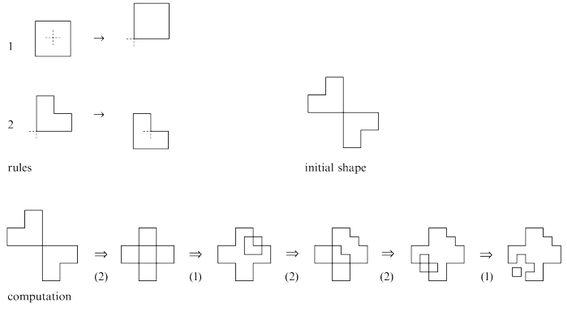


Figura 2.3_1 Una shape grammar composta da due regole e una forma iniziale e un calcolo di forma con le regole.
(Fonte: Knight, T. (2003). "Computing with emergence")

La figura in alto mostra una grammatica composta da due regole e una forma iniziale. La prima regola sposta un quadrato lungo la sua diagonale. La seconda regola sposta una forma a L, anch'essa lungo la diagonale. La forma iniziale è composta da due forme a L.
Il calcolo mostrato in figura è solo uno dei tanti calcoli possibili usando le due regole. Come abbiamo visto nel primo capitolo infatti, sono possibili svariati tipi di calcoli perché le regole sono non deterministiche, ovvero le loro applicazioni richiedono scelte da parte di chi usa la grammatica, sia esso umano o macchina. Ad ogni passaggio di un calcolo, ci possono essere tre tipi di scelte: (1) quale regola applicare, (2) a quale forma il lato sinistro di una regola è abbinato e (3) quale trasformazione usare per abbinare il lato sinistro di una regola a una forma. La gamma di scelte offerte dalle regole di una grammatica può determinare, in parte, il numero di forme che possono essere calcolate.

Una forma emergente è una forma, o parte di essa, che non è determinata dall'applicazione di una regola in una fase precedente del calcolo. Cioè, non è una forma τ(B) aggiunta da una precedente applicazione di una regola A→B. In shape grammar infatti, una forma può essere suddivisa in modi illimitati, sia essa composta da linee, piani o solidi. In altre parole, le forme non hanno primitive fisse che limitano i modi con cui possono essere calcolate ma, come sottolineato da Stiny[1], sono ambigue.
Poiché le forme sono liberamente scomponibili, le forme emergenti possono essere riconosciute in qualsiasi fase del calcolo e possono essere illimitate non solo in numero ma anche in natura: una forma emergente può essere infatti la somma di forme risultanti da precedenti applicazioni di regole, di parti di queste o può essere una parte di una singola forma.
In figura 2.3_2 è specificato come si manifesta l'emergenza nel processo illustrato nella figura 2.3_1: applicando la seconda regola alla forma iniziale si ottiene una sovrapposizione delle due L da cui emerge un quadrato; applicando a questo la prima regola emerge una L e così via...

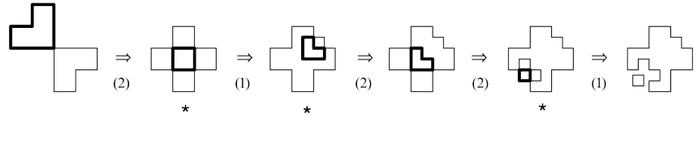


Figura 2.3_2 Calcolo della forma illustrata in figura 2.3_1. Le linee in grassetto indicano la forma a cui viene abbinata una regola in ogni passaggio. Gli asterischi identificano le forme emergenti.
(Fonte: Knight, T. (2003). "Computing with emergence")

Come abbiamo anticipato nel primo capitolo, in shape grammar l'emergenza varia da completamente prevedibile a completamente imprevedibile. Knight riconosce tre tipi di emergenze fondamentali: anticipata, possibile e imprevista.

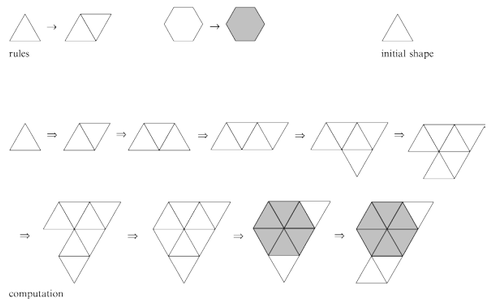
Emergenza anticipata

Nell'emergenza anticipata, l'autore della grammatica è in grado di prevedere che certe forme emergeranno dall'applicazione delle regole. Può pertanto includere nelle grammatica le regole che opereranno su tali forme emergenti. La Figura 2.3_3 illustra l'emergenza anticipata. Viene fornita una grammatica con due regole. La prima regola concatena due triangoli equilateri. È facile prevedere che applicando in modo ricorsivo la regola verrà a formarsi un esagono emergente. Sapendo questo, è possibile includere nelle grammatica una seconda regola che si applica all'esagono. Viene pertanto mostrato un calcolo che inizia con un singolo triangolo. Dopo diverse applicazioni della prima regola, viene creato un esagono a cui viene applicata la seconda regola, che prevede la campitura dello stesso.

1 Knight, T. (2003). "Computing with emergence". Environment and Planning B: planning and design, 30(1), 125-155. I materiali di questo paragrafo sono tratti prevalentemente da questo testo.

1 Stiny, G. (2001). "How to calculate with shapes", in Formal Engineering Design Synthesis, Eds. E. K. Antonsson, J. Cagan, Cambridge University Press, New York, 20-64.

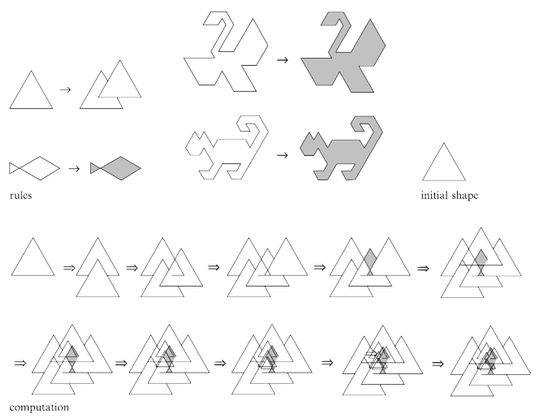
Figura 2.3_3 Emergenza anticipata.
(Fonte: Knight, T. (2003). "Computing with emergence")



Emergenza possibile

Nell'emergenza possibile, l'autore della grammatica ipotizza che determinate forme possano emergere, ma non può dirlo con certezza. Vengono quindi realizzati "piani di emergenza" che includono regole da applicare a eventuali forme emergenti. La Figura 2.3_4 illustra questo concetto. Viene fornita una grammatica con quattro regole. La prima regola sovrappone un triangolo a un altro esistente. Applicando ricorsivamente questa regola emerge, al quarto passaggio, una forma che è facile prevedere che emergerà ancora nel corso del processo generativo. Viene pertanto inclusa nel set di regole (regola 2). Diverse altre forme potrebbero emergere da questo processo, come ad esempio quelle incluse nelle regole 3 e 4. Tuttavia, in questo caso, non è possibile dire con certezza se queste regole verranno mai applicate.

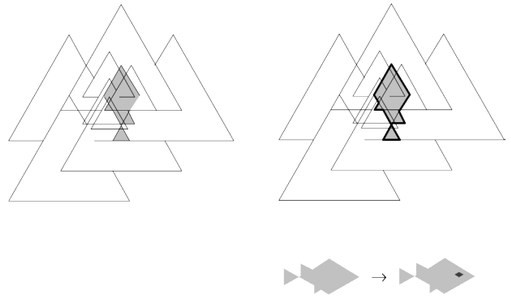
Figura 2.3_4 Emergenza possibile
(Fonte: Knight, T. (2003). "Computing with emergence")



Emergenza imprevista

Nell'emergenza imprevista infine, le forme emergenti non sono previste o anticipate in alcun modo.
La Figura 2.3_5 illustra un'emergenza imprevista. L'ultima forma nel calcolo nella figura 2.3_4 è mostrata ingrandita a sinistra. La forma emergente, indicata in grassetto nella forma a destra, è una variante più complessa di quella integrata nella regola 2. Per calcolarla, è possibile aggiungere una nuova regola alla grammatica.

Figura 5 Emergenza imprevista
(Fonte: Knight, T. (2003). "Computing with emergence")



Questa possibilità di aggiornare la grammatica con nuove regole rappresenta proprio quella trasformazione dello spazio concettuale di esplorazione di cui si è discusso nel capitolo precedente.
Pertanto, l'emergenza imprevista è la chiave di molte applicazioni di shape grammar per la ricerca di soluzioni progettuali originali. Questo tipo di emergenza del resto caratterizza un processo creativo di progettazione convenzionale, cioè non computazionale, in particolare nelle prime fasi concettuali della progettazione quando è importante l'esplorazione aperta di idee progettuali nuove e inaspettate.

Va notato comunque che l'emergenza inaspettata risulta indesiderata nelle applicazioni di shape grammar per l'analisi di un'architettura o di uno stile. In questo caso è sicuramente da prediligersi l'emergenza anticipata in quanto, in un lavoro di analisi, è necessario infatti che l'emergenza sia prevista nelle regole, in modo tale da generare solo una gamma accuratamente circoscritta di nuove forme coerenti con lo stile analizzato.

CAPITOLO 3

SHAPE GRAMMAR COME STRUMENTO DI ANALISI

In questo capitolo verrà descritto l'uso di shape grammar per l'analisi della struttura compositiva dell'opera di architetti o di stili architettonici che verrà illustrato attraverso lo studio di due ricerche paradigmatiche su shape grammar, riportate in forma di schede di approfondimento: la Palladian Grammar di Stiny e Mitchell e il Prairie Language di Koning e Eisenberg.

3.1_Derivazione delle regole compositive

Background

Per i primi vent'anni dalla sua invenzione, shape grammar è stata utilizzata principalmente come strumento di analisi. La sua prima applicazione per uno studio analitico è data da Stiny nel suo articolo del 1977, "Iceray: a note on the generation of Chinese lattice designs"[1]: con cinque semplici regole, la grammatica è in grado di catturare le convenzioni compositive dei pattern reticolari tipici dei motivi ornamentali tradizionali cinesi e generare pattern simili, sia esistenti che ipotetici, nello stesso stile[2].

Questo studio ha definito gli standard per le successive ricerche e la pubblicazione della palladian grammar[3] di Stiny e Mitchell nel 1978 diede inizio ad una serie di ricerche sull'uso di shape grammar per lo studio degli stili architettonici[4]. Negli anni seguenti, la grammatica analitica è stata ampiamente utilizzata in numerosi lavori, definendo strategie generali e creando una base di conoscenze per comprendere lo stile compositivo di un particolare architetto[5]. Negli anni '80 e '90, shape grammar è stata utilizzata per analizzare le opere di Giuseppe Terragni[6], Frank Lloyd Wright[7], Glenn Murcutt[8] e Christopher Wren[9], nonché per gli stili vernacolari delle sale da tè giapponesi[10], delle case tradizionali taiwanesi[11] e per l'architettura del paesaggio dei giardini Mughul[12].

---

1 Stiny, G. (1977). "Ice-Ray: A Note on the Generation of Chinese Lattice Designs", Environment and Planning B 4, no. 1, 89-98.

2 Knight, T. (1999). "Applications in architectural design, and education and practice". Report for the NSF. In MIT Workshop on Shape Computation.

3 Stiny, G. & Mitchell, W.J. (1978). "The Palladian grammar," Environment and Planning B 5, 5-18. Per un approfondimento su questo lavoro vedi Scheda 1.

4 Knight, T. (1999). "Applications in architectural design, and education and practice". Report for the NSF. In MIT Workshop on Shape Computation.

5 Tepavčević, B., & Stojaković, V. (2012). "Shape grammar in contemporary architectural theory and design", Facta Universitatis-series: Architecture and Civil Engineering, 10(2), 169-178.

6 Flemming, U. (1981) "The secret of the Casa Guiliani Frigerio," Environment and Planning B 8, 87-96.

7 Koning, H. & Eisenberg, J. (1981) "The language of the prairie: Frank Lloyd Wright's prairie houses", Environment and Planning B 8, 295-323.

8 Hanson, N.L.R. & Radford, A.D. (1986). "On Modelling the Work of the Architect Glenn Murcutt", Design Computing, 189-203.

9 Buelinckx, H. (1993) "Wren's language of City church designs: a formal generative classification", Environment and Planning B: Planning and Design 20, 645-676.

10 Knight, T. (1981) "The forty-one steps," Environment and Planning B 8, 97-114.

11 Chiou, S-C & Krishnamurti, R. (1995). "The grammar of Taiwanese traditional vernacular dwellings", Environment and Planning B: Planning and Design 22, 689-720.

12 Stiny, G. & Mitchell, W.J. (1980). "The grammar of paradise on the generation of Mughul gar-

Derivazione delle regole compositive

La definizione di una grammatica per la derivazione delle regole compositive di uno stile architettonico implica essenzialmente i seguenti passaggi: (1) si individuano un vocabolario di forme e un insieme di relazioni spaziali comuni all'interno del corpo d'opere analizzato; (2) vengono definite le regole di forma che fissano tali relazioni spaziali; (3) viene indicata una forma iniziale da cui far partire il processo generativo; e (4) viene specificata la grammatica per quello stile[1].
Appare chiaro dunque che l'uso di shape grammar non serve a ricostruire automaticamente in modo univoco e oggettivo lo stile analizzato, l'interpretazione dell'autore della grammatica ha un ruolo chiave:

"The shape grammar generates the original corpus of designs as well as new designs in the same style. Different interpretations of the composition of designs in a certain style are always possible. These may lead to different vocabularies and spatial relations, different grammars and different languages. Each of these different grammars and languages provides a different understanding, or theory, of a style" [2].

L'obiettivo principale di una grammatica analitica infatti non risiede tanto nella modellazione di edifici geometricamente e storicamente accurati, ma nella realizzazione di modelli qualitativamente corretti che definiscono relazioni di dipendenza complesse tra elementi architettonici. In questo senso, shape grammar permette un'analisi architettonica su livelli più complessi di quella tradizionale, che non potrebbero essere realizzati senza l'appropriato background computazionale[3].
Il caso studio a fine capitolo, che mette a confronto le analisi fatte da Eisenman (non computazionale) e Flemming (che invece utilizza shape grammar) di Casa Giuliani Frigerio di Terragni, metterà in luce proprio la capacità di shape grammar di rilevare pattern compositivi difficilmente rintracciabili con metodi tradizionali.

dens", Environment and Planning B 7, 209-226.

1 Knight, T. W. (1993). "Color grammars: the representation of form and color in designs", Leonardo, 117-124: 117.

2 Ibid. Traduzione: "La grammatica è in grado di generare tanto il corpus originale quanto nuove soluzioni nello stesso stile. Sono sempre possibili diverse interpretazioni della composizione dei disegni in un certo stile. Questi possono portare a differenti vocabolari e relazioni spaziali, differenti grammatiche e linguaggi. Ciascuna di queste diverse grammatiche e lingue fornisce una diversa interpretazione dello stile".

3 Tepavčević, B., & Stojaković, V. (2012). "Shape grammar in contemporary architectural theory and design", Facta Universitatis-series: Architecture and Civil Engineering, 10(2), 169-178.

Scheda 1: Palladian Grammar
generazione automatica di piante architettoniche

Nell'articolo "The palladian grammar"[1], Stiny e Mitchel hanno proposto un metodo basato su una shape grammar parametrica per la generazione delle piante delle ville di Palladio. Le regole grammaticali, basate sugli esempi di piante riportate nei Quattro Libri dell'Architettura, rileggono il sistema di proporzioni e il linguaggio architettonico di Palladio in forma "generativa". Seguendo l'interpretazione di Wittkower[2], che vedeva nella composizione in pianta la caratteristica distintiva delle ville palladiane, altri aspetti del sistema architettonico, come prospetti e elementi decorativi, non furono considerati nel loro lavoro.
Nell'articolo hanno utilizzato tale metodo per creare la pianta di villa Malcontenta.

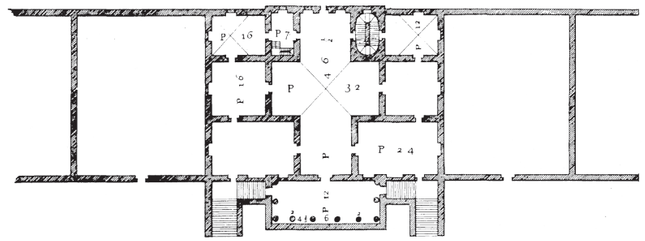


Figura 3.2_1 La Villa Malcontenta disegnata da Palladio nei Quattro Libri.

Innanzi tutto, gli autori notano che le piante di tutte tranne due delle ville del Palladio (tavole XIII e XLIII) nei Quattro Libri sono definite rispetto a un singolo asse. Pertanto, la forma iniziale da cui tutte le piante sono generate dall'applicazione sequenziale e ricorsiva delle regole specificate di seguito è la forma etichettata

1 Stiny, G., & Mitchell, W. J. (1978). "The palladian grammar", Environment and planning B: Planning and design, 5(1), 5-18. I materiali di questa scheda sono tratti da questo testo.

2 Wittkower, R. (1952). Architectural Principles in the Age of Humanism, Alec Tiranti, London.

(s_Φ, {(0, 0): A}) mostrata nella figura 3.2_2. La forma iniziale implica che i piani siano posizionati all'origine del sistema di coordinate.

Il processo di generazione delle piante avviene quindi in otto fasi che corrispondono più o meno a un processo di progettazione naturale e intuitivo. Le fasi vengono applicate in questa sequenza:
(1) definizione della griglia;
(2) definizione dei muri esterni;
(3) layout delle stanze;
(4) riallineamento delle pareti interne;
(5) ingressi principali: portici e inflessioni delle pareti esterne;
(6) ornamenti esterni: colonne;
(7) finestre e porte;
(8) risoluzione.

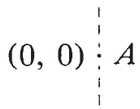
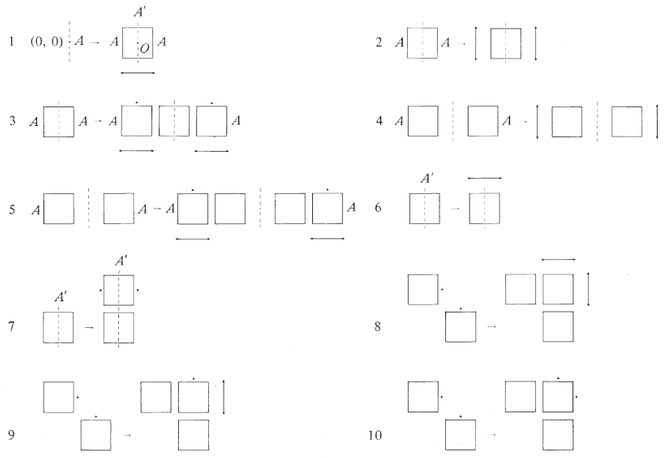


Figura 3.2_2 La forma iniziale da cui vengono generate tutte le piante della villa.

Fase 1: definizione della griglia

Le piante sono costruite in termini di griglie rettangolari etichettate con simmetria bilaterale rispetto all'asse nord-sud del sistema di coordinate. Le griglie sono generate utilizzando le regole specificate nella figura 3.2_3. La griglia necessaria per il layout della pianta di Villa Malcontenta è mostrata nella figura 3.2_4.

Figura 3.2_3 Regole per la generazione di griglie con simmetria bilaterale. Le griglie generate da queste regole vengono utilizzate per definire la struttura compositiva delle piante.



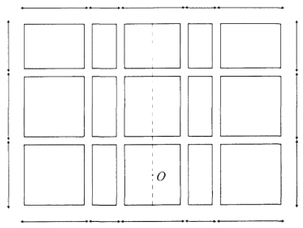


Figura 3.2_4 La griglia sottostante generata per Villa Malcontenta.

Fase 2: definizione della parete esterna

Una volta generata una griglia, viene circoscritta da un rettangolo per formare un muro esterno. Questa operazione viene eseguita dalla regola specificata nella figura 3.2_5. Il modello di parete sottostante generato per Villa Malcontenta applicando questa regola alla griglia nella figura 3.2_4 è mostrato nella figura 3.2_6.

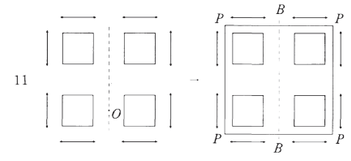


Figura 3.2_5 Regola per la generazione di pareti esterne.

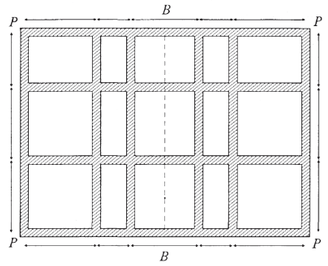


Figura 3.2_6 Il pattern sottostante le pareti di Villa Malcontenta.

Fase 3: layout delle stanze

Gli spazi interni delle piante uniassiali di Palladio possono essere rettangolari, a forma di T, a doppio T o a croce. Queste forme si ottengono concatenando ricorsivamente gli spazi generati nelle fase 2, in conformità con le regole specificate nella figura 3.2_7. Le applicazioni di queste regole preservano la simmetria della pianta.

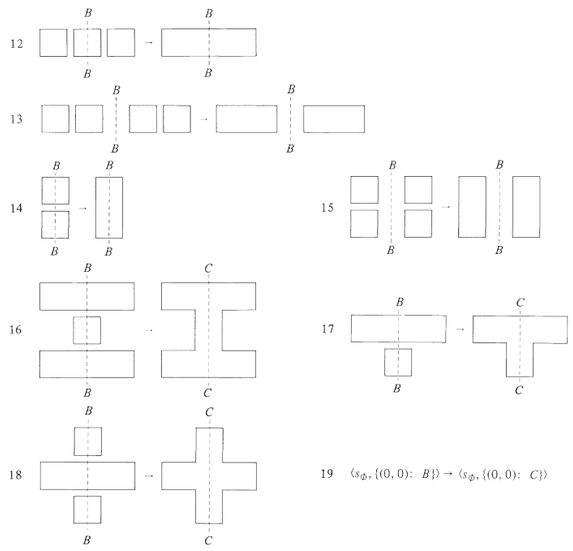


Figura 3.2_7 Regole la definizione del layout delle stanze.

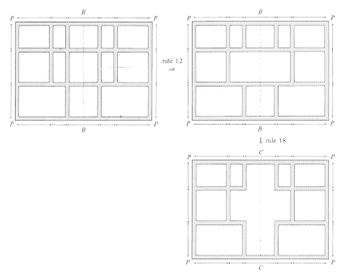


Figura 3.2_8 Generazione del layout delle stanze di Villa Malcontenta.

Si noti che un piano può avere al massimo uno spazio non rettangolare e che questo spazio deve essere diviso in due dall'asse nord-sud del sistema di coordinate.
La generazione del layout delle stanze per Villa Malcontenta è data nella figura 3.2_8. Il primo disegno in questa figura è prodotto applicando la regola 13 allo schema in figura 3.2_6.

Fase 4: riallineamento delle pareti interne

Stiny e Mitchel notano che Palladio talvolta fa ricorso a spostamenti di pareti interne in direzione nord-sud ed est-ovest per apportare lievi modifiche al layout delle stanze. Le regole per i riallineamenti sono specificate nella figura 3.2_9 e vengono applicate ai layout delle stanze generati nella fase 3. L'applicazione di queste regole preserva la simmetria del piano. Il riallineamento del muro è un'operazione opzionale, è usato molto raramente.

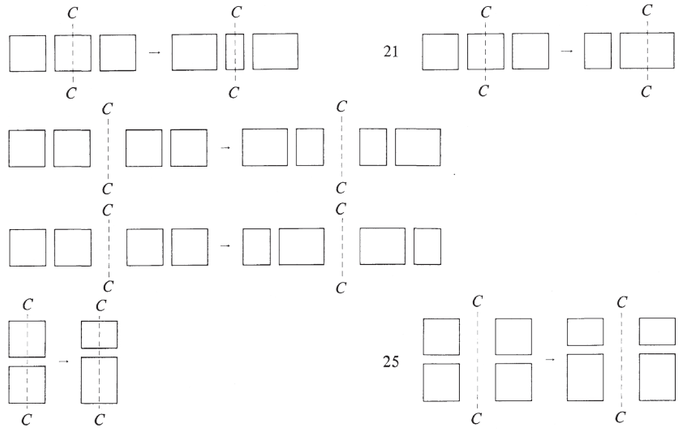


Figura 3.2_9 Regole per riallineare le pareti interne.

Fase 5: ingressi principali

Le regole di cui alla figura 3.2_10 specificano la definizione degli ingressi principali. Questi ingressi si trovano sull'asse di simmetria del piano. Uno si distingue per la presenza di un portico in antis (regole 27-34) o pròstilo (regole 35-39); l'altro si distingue per un portico in antis (regole 40-45), una flessione della parete esterna dello spessore di mezza parete (regola 46), o niente (regole 49 e 50). L'aggiunta di un portico e un'inflessione del muro al piano di Villa Malcontenta è mostrata nella figura 3.2_11. Il primo disegno nella figura 3.2_11 è prodotto applicando la regola 26 all'ultimo disegno nella figura 3.2_8.

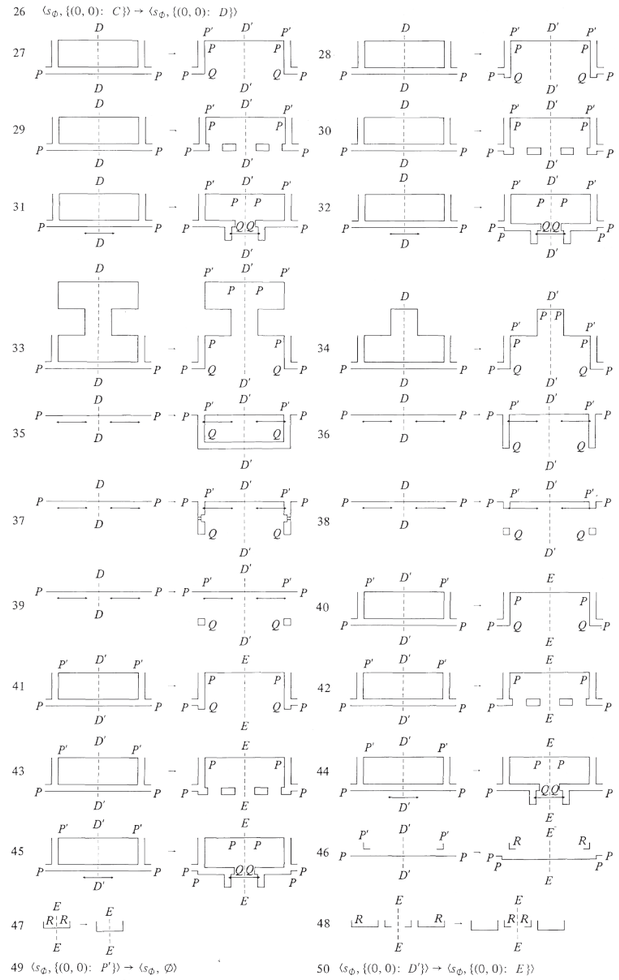


Figura 3.2_10 Regole per l'aggiunta di portici e inflessioni delle pareti agli ingressi principali di una villa.

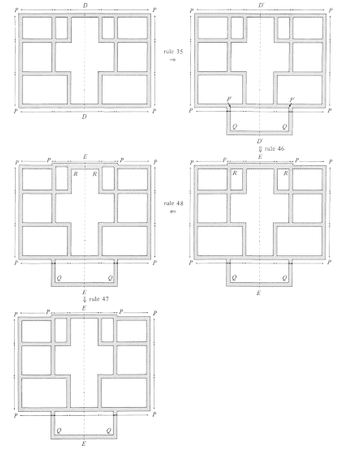


Figura 3.2_11 Generazione del portico e inflessione del muro per Villa Malcontenta.

Fase 6: ornamenti esterni

Le regole indicate nella figura 3.2_12 specificano i diversi modi in cui è possibile aggiungere le colonne per completare i portici. L'inserimento della colonna per Villa Malcontenta è mostrato nella figura 3.2_13. Questo disegno è prodotto dall'ultimo disegno nella figura 3.2_11 applicando la regola 56.

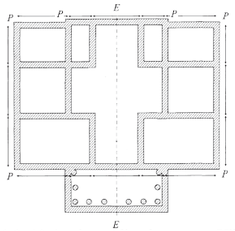


Figura 3.2_13 Aggiunta di colonne per Villa Malcontenta.

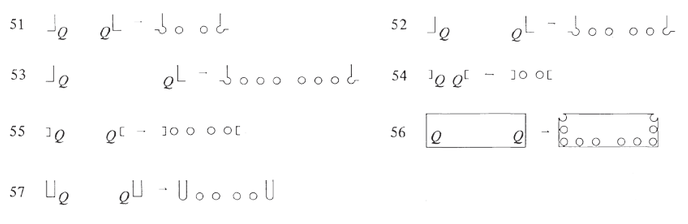


Figura 3.2_12 Regole per la finitura dei portici mediante l'aggiunta di colonne.

Fase 7: finestre e porte

Le regole 58 e 59 specificate nella figura 3.2_14 fissano le condizioni per cui le finestre possono essere inserite in pianta. Le regole 60 e 61 consentono di posizionare simmetricamente le porte interne non assiali in modo da allinearle con le finestre precedentemente inserite e tra loro. Le regole 62-65 prevedono che le porte esterne siano posizionate sull'asse di simmetria del piano.

L'inserimento di finestre e porte in Villa Malcontenta è mostrato nella figura 3.2_15. Il primo disegno in questa figura è ottenuto dal disegno in figura 3.2_13 applicando la regola 58.

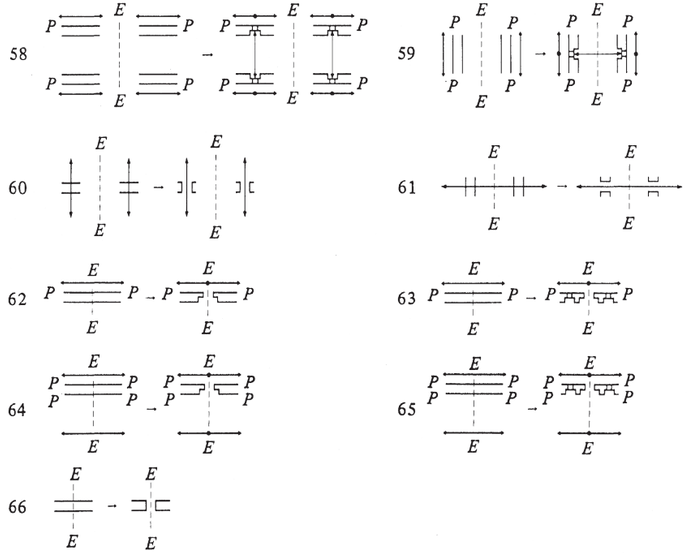


Figura 3.2_14 Regole per localizzare finestre e porte nei piani della villa.

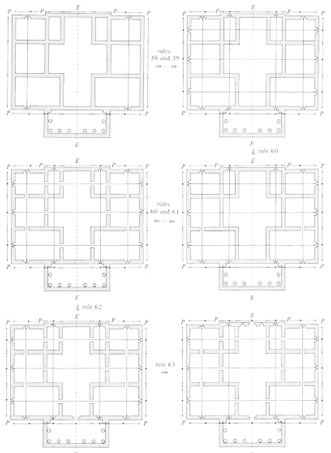


Figura 3.2_15 Generazione di finestre e porte in Villa Malcontenta.

Fase 8: risoluzione

Le regole per terminare il processo sono specificate nella figura 3.2_16. Queste regole prevedono la cancellazione delle etichette e dei segmenti di linea etichettati utilizzati per guidare il processo generativo. La pianta finale di Villa Malcontenta è mostrata nella figura 3.2_17.

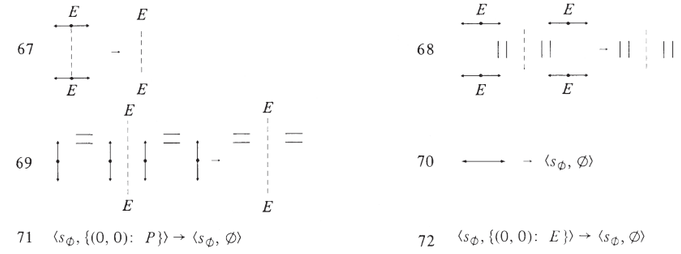


Figura 3.2_16 Regole di risoluzione.

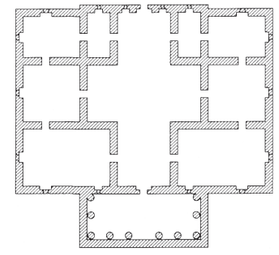


Figura 3.2_17 Il piano definitivo generato per Villa Malcontenta.

Scheda 2: Prairie language
grammatica tridimensionale

Nell'articolo "The language of the prairie: Frank Lloyd Wright's prairie houses"[1], Koning e Eisenberg mostrano come la composizione delle prairie houses di Wright si basi su alcune semplici relazioni spaziali tra volumi. Queste relazioni spaziali forniscono la base per una shape grammar parametrica che stabilisce una struttura ricorsiva dalla quale possono essere generati tanto le case progettate da Wright quanto nuovi progetti nello stesso stile. Questo lavoro è stato particolarmente importante per le successive implementazioni di shape grammar, soprattutto come strumento progettuale, in quanto è stata la prima grammatica architettonica tridimensionale, ispirata in parte al precedente lavoro di Stiny sulle grammatiche del kindergarden[2] e alla possibile influenza di Froebel sull'architettura di Wright[3].

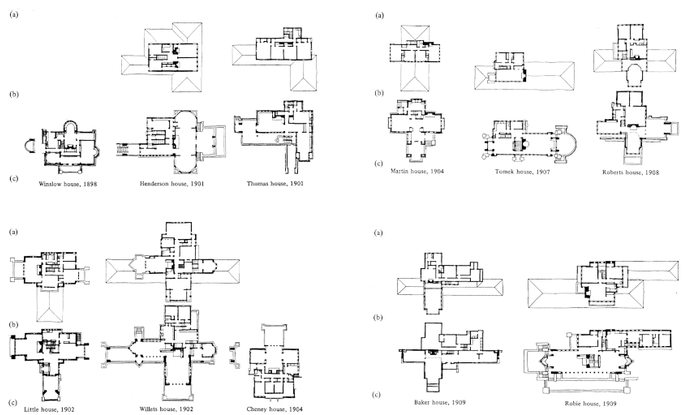


Figura 3.3_1 Il corpus: undici prairie houses progettate da Frank Lloyd Wright

1 Koning, H., & Eizenberg, J. (1981). "The language of the prairie: Frank Lloyd Wright's prairie houses", Environment and planning B: planning and design, 8(3), 295-323. I materiali di questa scheda sono tratti da questo testo.

2 Stiny, G. (1980). "Kindergarten grammars: designing with Froebel's building gifts", Environment and Planning B: Planning and Design, 7(4), 409-462.

3 Koning, H. & Eisenberg, J. (1981) "The language of the prairie: Frank Lloyd Wright's prairie houses"; Knight, T. (1999). "Applications in architectural design, and education and practice", Report for NSF/MIT workshop on shape computation, NSF/MIT workshop on shape computation; MacCormac, R.C. (1974). "Froebel's kindergarten gifts and the early work of Frank Lloyd Wright", Environment and Planning B 1, 29-50; Manson, G. (1958). Frank Lloyd Wright to 1910, Reinhold, New York.

Prairie grammar

La grammatica si basa su un corpus di undici progetti di Wright mostrati nella figura 3.3_1. Per facilitare la definizione della grammatica gli autori hanno tradotto i disegni di Wright negli schemi volumetrici mostrati in figura 3.3_2.

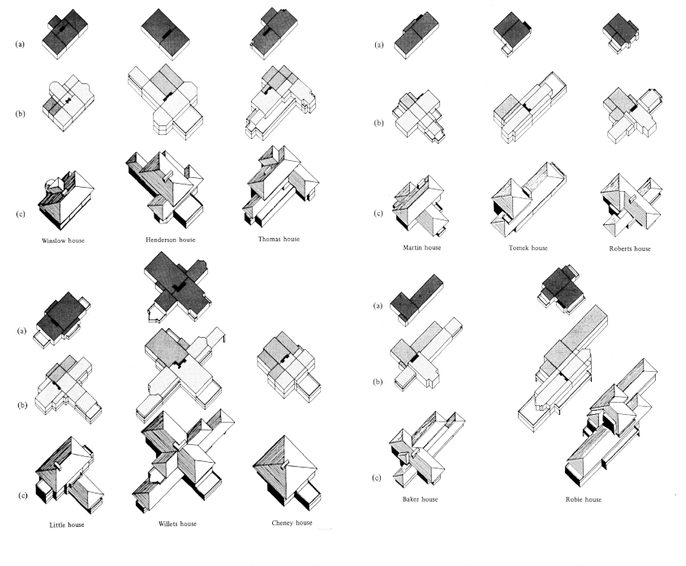


Figura 3.3_2 Le case di figura 3.3_1, stilizzate e ridotte a quattro zone funzionali: soggiorno, servizio, veranda, camera da letto. In (a), viene mostrato il livello della camera da letto; in b) il livello del piano principale; e in (c) l'esterno.

In questi schemi sono mostrati solo gli elementi compositivi principali, secondo le prescrizioni fornite dallo stesso Wright: il caminetto viene pertanto individuato come il fulcro della composizione intorno al quale si aggregano i volumi che definiscono i vari ambienti.

I principali elementi del vocabolario della grammatica sono blocchi rettangolari tridimensionali, parametrizzati come mostrato in figura 3.3_3. In particolare, l'altezza dei blocchi è fissa, mentre la loro lunghezza e larghezza sono variabili. Le figure 3.3_4 e 3.3_5 mostrano le convenzioni grafiche adottate.

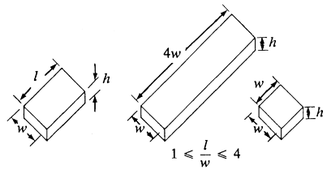


Figura 3.3_3 Parametri per gli elementi del vocabolario principale

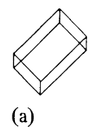


Figura 3.3_4 Convenzioni grafiche per linee nascoste,

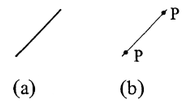


Figura 3.3_5 Convenzioni grafiche per le linee perimetrali nei disegni: la linea a doppio spessore (a) corrisponde alla linea etichettata (b).

Regole di forma e composizioni di base

Il processo progettuale viene codificato ed elaborato negli schemi di regole di forma 1-18 specificati nella figura 3.3_6.

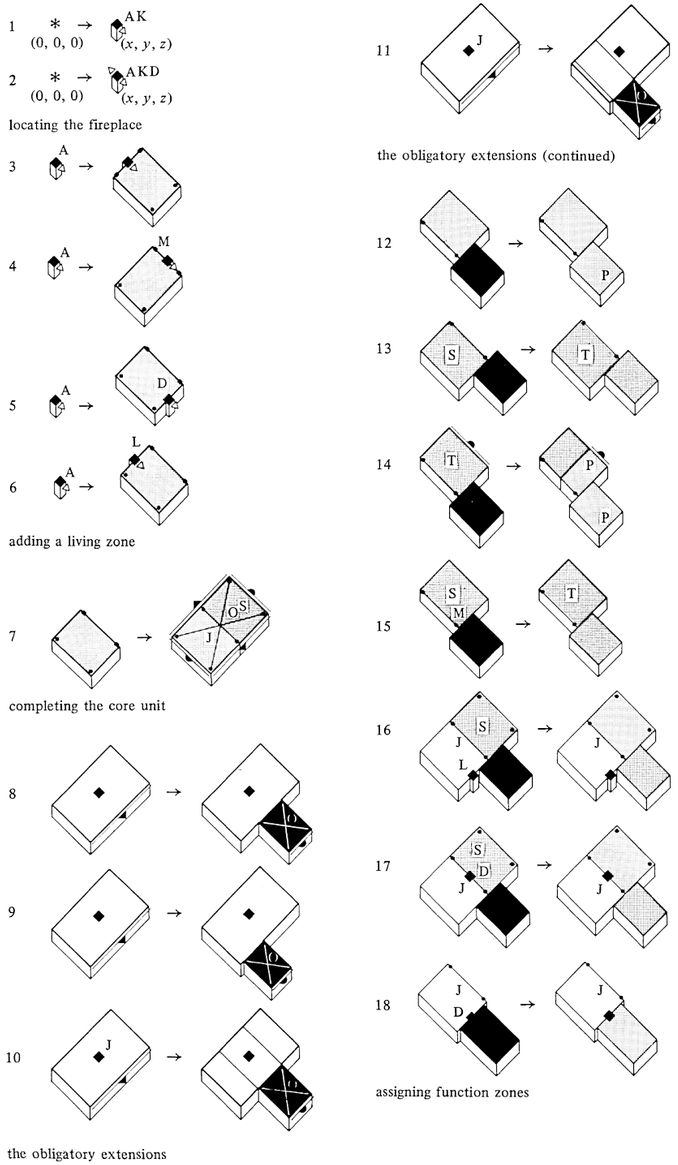


Figura 3.3_6 Schemi delle regole di forma della composizione di base.

Gli schemi 1 e 2 si applicano alla forma iniziale della grammatica, costituita dal simbolo * associato al punto (0, 0, 0), per posizionare il caminetto. Lo schema 1 produce un camino a focolare singolo, lo schema 2 a camino a doppio focolare. Una volta posizionato il caminetto, è possibile aggiungere una zona giorno in uno dei quattro modi determinati dagli schemi 3-6. Una zona di servizio viene quindi aggiunta alla zona giorno mediante un'applicazione dello schema 7. In questo modo, viene fissata l'unità centrale della casa.

Nelle figure 3.3_7-9 è mostrato il processo generativo, mentre una rappresentazione schematica di alcune sequenze ammissibili di applicazioni di questi schemi è riportata nella figura 3.3_10. In totale, gli schemi 1-18 possono essere utilizzati per generare ottantanove composizioni (figura 3.3_11).

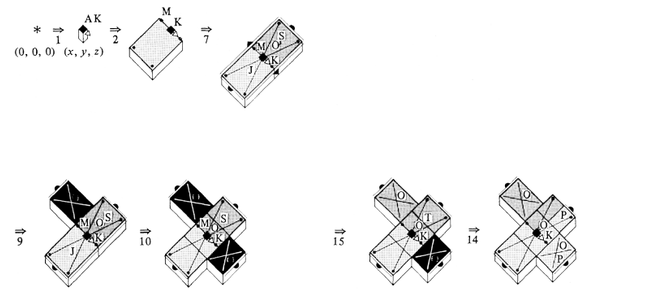


Figure 3.3_7-8-9 Esempi di processo generativo. Gli schemi delle regole di forma vengono applicati alla forma iniziale per stabilire un'unità centrale. I numeri sotto le frecce indicano gli schemi applicati.

Composizioni complete

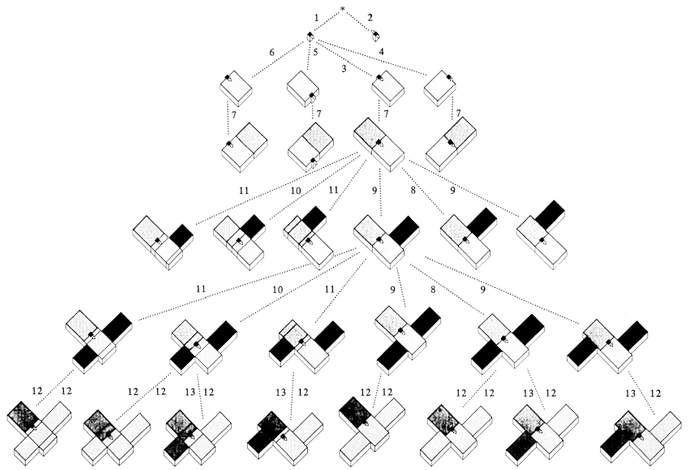


Figure 3.3_10 Possibili sequenze ammissibili di applicazioni degli schemi rappresentati in figura 3.3_6.

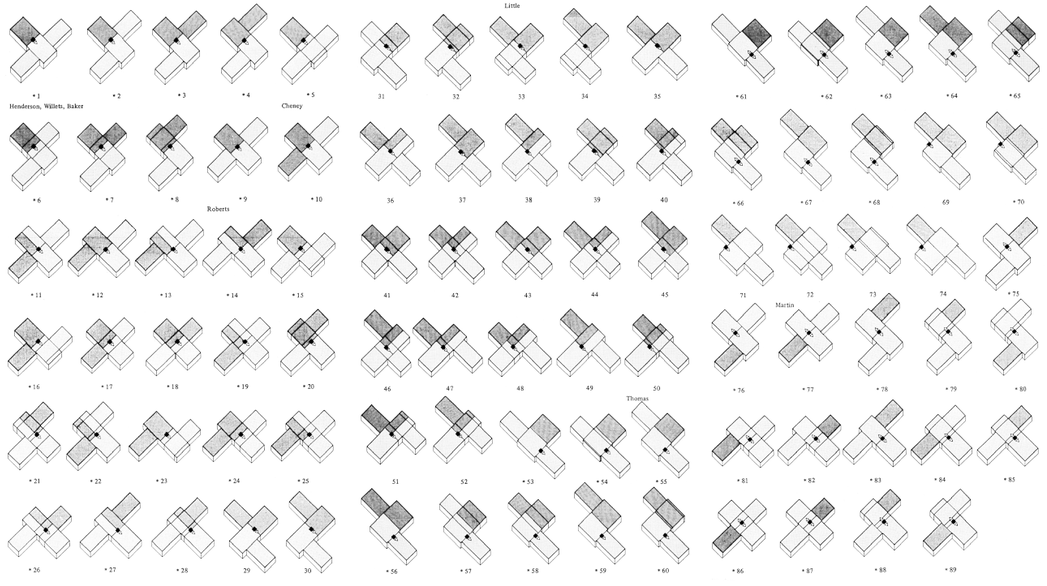


Figure 3.3_11 Le ottantanove composizioni di base possibili

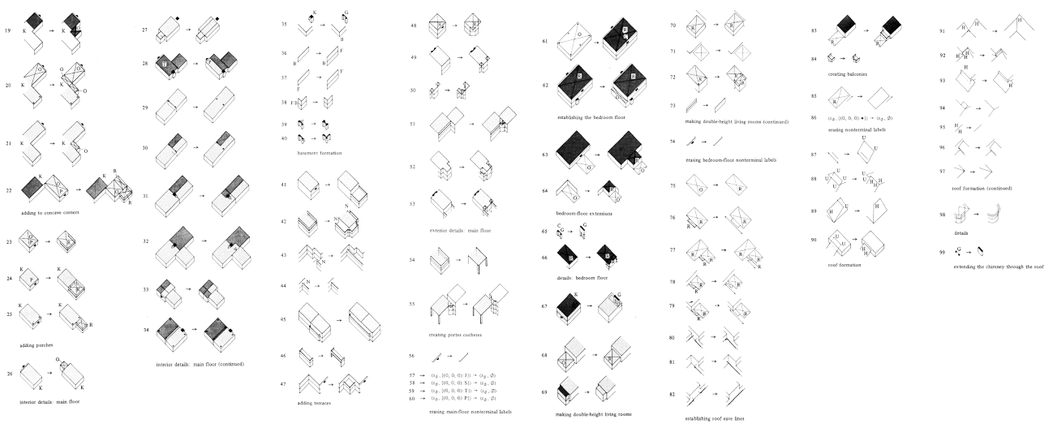


Figure 3.3_12 Regole per avere le composizioni complete

Le composizioni di base elencate nella figura 3.3_11 possono essere elaborate in vari modi per produrre progetti completi attraverso l'applicazione delle regole specificate nella figura 3.3_12.

Alcuni di questi schemi, ad esempio quelli che aggiungono tetti o basamenti, si applicano obbligatoriamente per ottenere i disegni finali dalle composizioni di base; altri, ad esempio, quelli che aggiungono portici, terrazze o blocchi extra agli angoli,

vengono utilizzati opzionalmente, consentendo così alle singole composizioni di base di portare a una serie di variazioni del design. L'applicazione degli schemi è illustrata nelle figure 3.3_13-17.

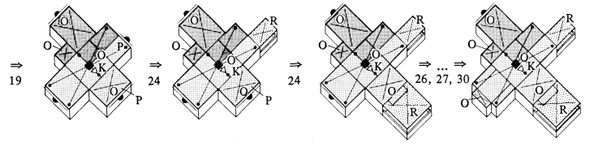


Figura 3.3_13

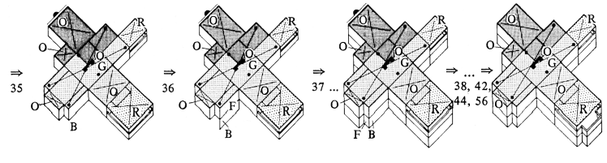


Figura 3.3_14

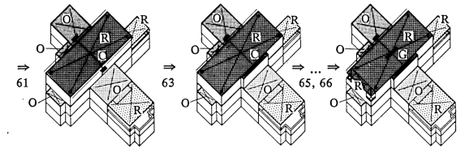


Figura 3.3_15

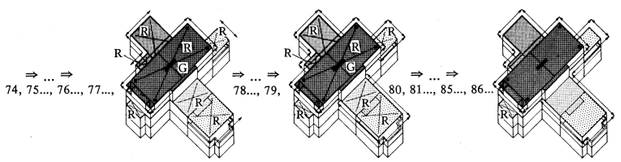


Figura 3.3_16

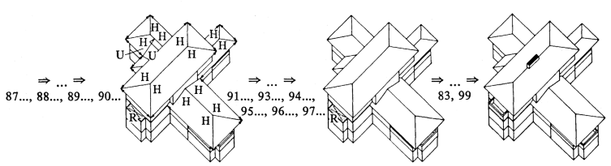


Figura 3.3_17

Case a due livelli e a doppia altezza

Nel catalogo della figura 3.3_18 sono riportati i progetti a due piani derivati dalle composizioni di base della figura 3.3_11 applicando il minor numero di schemi necessari per produrre un progetto complete. Per ciascuno dei disegni nella figura 3.3_18 esiste un corrispondente disegno a un solo piano. Inoltre, per quelle composizioni di base contrassegnate da un asterisco nella figura 3.3_11, è anche possibile realizzare progetti di spazi abitativi a doppia altezza.

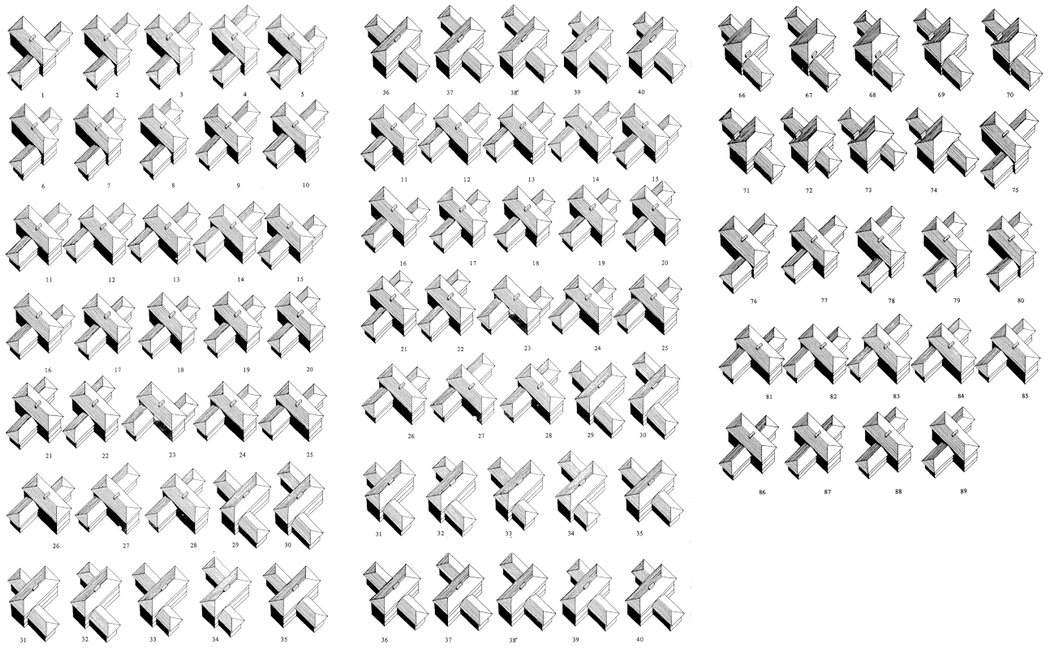


Figura 3.3_18

La Figura 3.3_19 mostra i tre tipi di alternative progettuali che possono essere derivate dalle composizioni di base. Pertanto, esiste un totale di 238 progetti nel linguaggio definito dalla grammatica sviluppata da Koning e Eisenberg che può essere ottenuta dalle composizioni di base della figura 3.3_11 senza applicare schemi di elementi aggiuntivi opzionali. Naturalmente, l'applicazione di questi schemi opzionali aumenta notevolmente il numero di soluzioni possibili.

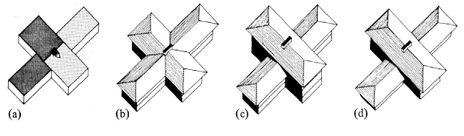


Figura 3.3_19

Progetti originali

La capacità della grammatica di generare progetti originali coerenti con la tecnica compositiva wrightiana è dimostrata attraverso tre composizioni illustrate nella figura 3.3_20.

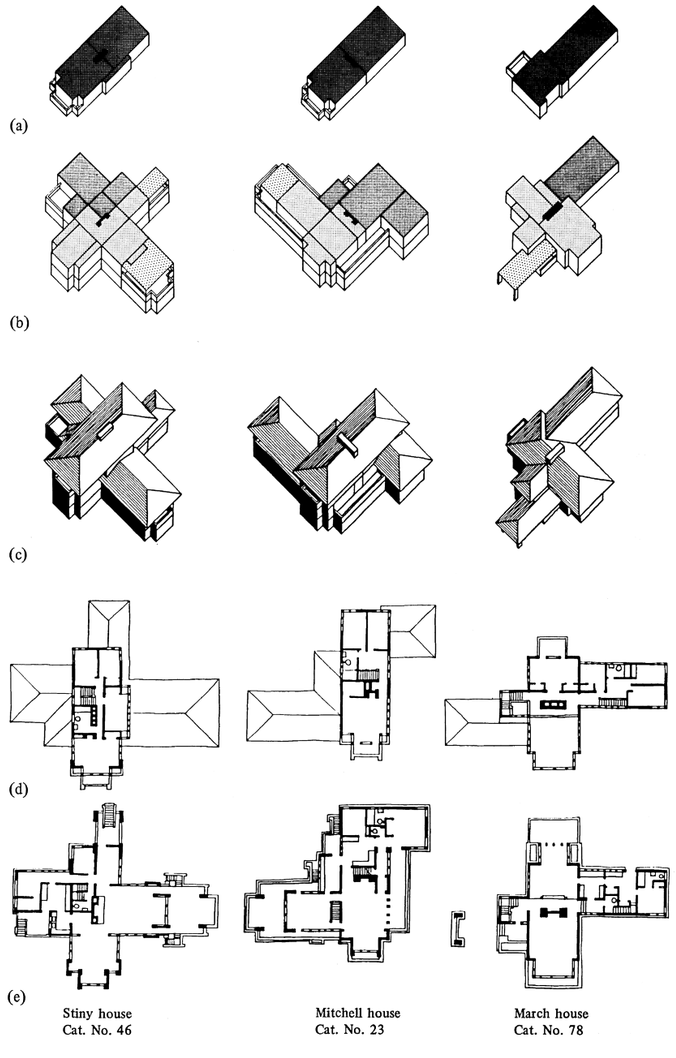


Figura 3.3_20 Tre nuovi progetti generati dalla grammatica: (a) livello della camera da letto, (b) livello del piano principale, (c) esterno; e le piante: (d) pianta della camera da letto, (e) pianta del piano principale.

CAPITOLO 4

SHAPE GRAMMAR COME STRUMENTO PROGETTUALE

4.1. Generazione di soluzioni originali
Scheda 3: Malagueira Grammar
Scheda 4: CGA Shape

In questo capitolo vengono discusse le principali linee di ricerca sull'uso di shape grammar per la generazione di soluzioni progettuali originali. Nel primo paragrafo vengono introdotte tali ricerche, le piuù interessanti delle quali verranno approfondite nelle schede 3 e 4.
Nella scheda 3 viene approfondita la Discursive Grammar di J. Duarte (2001) per l'ampliamento del progetto di Siza a Malagueira, in quanto perfetto esempio di come soluzioni originali possano essere sviluppate a partire dall'analisi di opere esistenti.
Nella scheda 4 viene approfondita CGA Shape, una grammatica sviluppata da Muller e Parish (2006) presso l'ETH di Zurigo, utilizzata per la creazione di un software, City Engine (2008), capace di generare soluzioni progettuali.

4.1_Generazione di soluzioni originali

Dal noto al nuovo

Le shape grammars possono in teoria essere create da zero, ma in pratica, sono solitamente create a partire da linguaggi e progetti esistenti[1]. Dunque, è un approccio alla progettazione che muove dall'approccio analitico illustrato nel capitolo 3. A proporre formalmente questo metodo per la prima volta fu Knight nel 1981 con l'articolo "Languages of designs: from known to new" [2], e il lavoro di Koning e Eisenberg[3], illustrato nel capitolo precedente (scheda 2) e pubblicato nello stesso anno, rappresenta una prima esplorazione di questa possibilità. Questo approccio alla progettazione "originale" si basa sulla trasformazione di una grammatica desunta da un'opera o uno stile esistente:

"Languages are created by transforming the spatial relations underlying grammars for existing languages. In other words, a known style is first analyzed by inferring a grammar for it, the rules of the grammar are transformed, and then the transformed rules become the basis for a new grammar and style" [4].

Come ha osservato Knight, tale metodo può essere usato tanto per caratterizzare l'evoluzione storica di stili esistenti, quanto per sviluppare nuovi progetti[5]. All'inizio degli anni '90 questo metodo è stato utilizzato soprattutto per scopi didattici. Agli studenti veniva chiesto di utilizzare shape grammar per desumere le regole compositive di linguaggi architettonici esistenti e applicare a queste varie modifiche per generare nuovi linguaggi[6]. Negli anni successivi, questo approccio è stato ulteriormente approfondito per lo studio di shape grammar come strumento di progettazione generativa all'interno dei progetti di ricerca nel programma di dottorato "Design and Computation" del MIT[7]. Di particolare rilievo, in questo contesto, sono le ricerche condotte da Birgul Colakoglu e Jose Duarte:

Birgul Colakoglu ha esplorato l'applicazione di una shape grammar informale per nuovi progetti di case che portano le caratteristiche stilistiche di una tradizionale

1 Knight, T. (1999). "Applications in architectural design, and education and practice". Report for the NSF. In MIT Workshop on Shape Computation, 1-11: 2.

2 Knight, T. (1981) "Languages of designs: from known to new", Environment and Planning B 8, 213-238.

3 Koning, H., & Eizenberg, J. (1981). "The language of the prairie: Frank Lloyd Wright's prairie houses", Environment and planning B: planning and design, 8(3), 295-323.

4 Knight, T. (1999). "Applications in architectural design, and education and practice", Report for the NSF, MIT Workshop on Shape Computation, 2.

5 Ibid.

6 Ibid.

7 Ibid.

casa “Hayat” in stile ottomano nel XVIII e XIX secolo tipiche della città di Sarajevo. Le nuove forme delle case dovevano essere utilizzate per generare quelle che Colakoglu definisce “interpolazioni” del tipo esistente in un dato contesto architettonico, al fine di adattarle a uno stile di vita contemporaneo[1].
Jose Duarte ha sviluppato un sistema basato sui progetti abitativi di Malagueira di Alvaro Siza Vieira[2]. Il suo lavoro è stato particolarmente significativo perché ha proposto un modello di successo con la capacità di generare diverse abitazioni di massa non ripetitive. A questo lavoro è dedicata la scheda 3 di questa tesi.

Tecniche di suddivisione

In alcuni casi shape grammar è stata utilizzata per sviluppare tecniche di suddivisione per generare elementi costruttivi o di finitura.
Un esempio significativo è rappresentato dal lavoro di Lawrence Sass che ha introdotto un nuovo metodo per generare progetti per abitazioni in lastre di compensato utilizzando shape grammar e un processo di fabbricazione CNC[3]. Shape grammar viene utilizzata per suddividere un solido iniziale in componenti costruibili realizzati con una macchina da taglio CNC. Questo metodo non richiede particolari elementi di fissaggio per le connessioni tra i pannelli: shape grammar viene utilizzata infatti anche per definire i metodi di assemblaggio e il taglio degli elementi tenuti insieme solo per attrito[4].
Di grande interesse sono anche le sperimentazioni condotte da Gehry Partners. Queste applicazioni di shape grammar, dirette alla razionalizzazione delle superfici per soddisfare specifici requisiti di costruibilità, si basano sull’assunto che le superfici con qualunque curvatura possano essere approssimate da elementi piani, deformati in un intervallo limitato[5]. Inoltre, si presume che la dimensione degli elementi piani sia determinata da valori della curvatura gaussiana.[6] Un esempio significativo è l’Experience Music Project a Seattle, dove l’algoritmo di suddivisione utilizzato

1 Colakoglu, B. (2002). “An Informal Shape Grammars for Interpolation of Traditional Bosnian Hayat Houses in a Contemporary Context”, Generative Art and Design Conference, Milan, Italy, 15.1-15.9
2 Duarte, J.P. “Towards the mass customization of housing: the grammar of Siza’s houses at Malagueira”, Environment and Planning B: Planning and Design 32(3), 347-380.
3 Sass, L. (2005). “Wood frame Grammar: CAD scripting a wood frame house, CAAD Futures Conference, Vienna, 1-10.
4 Tepavčević, B. & Stojaković, V. (2012). “Shape Grammar in contemporary architectural theory and design”. Vedi anche: Özkar, M., & Stiny, G. (2009), “Shape grammars”. In ACM SIGGRAPH 2009 Courses, 1-176.
5 Shelden, D. (2002). *Digital Surface Representation and the Constructibility of Gehry’s Architecture*, PhD Disseration, MIT.
6 Ibid.

era derivato dalla grammatica di Mitchell e Stiny del “Mughul garden”[1]: la regola di base della grammatica suddivide un quadrato in quattro quadrati più piccoli e, a seconda della curvatura locale della superficie, può essere ricorsivamente applicata per ottenere regioni quadrate piane più piccole di varie dimensioni[2].

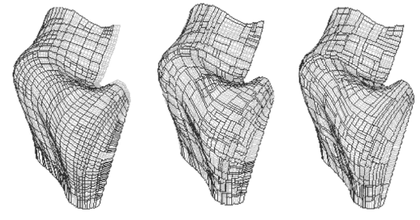


Figura 4.1_1 Grammatica di suddivisione applicata all'Experience Music Project di Gehry a Seattle.
(Fonte: Shelden, D. (2002). Digital Surface Representation and the Constructibility of Gehry’s Architecture).

Un altro esempio notevole è dato dal lavoro di Michael Hansmeyer che, dal 2010, ha condotto una serie di esperimenti sull’utilizzo di shape grammar per la suddivisione di superfici. I processi di suddivisione traggono ispirazione dall’idea di auto-somiglianza, che può essere ricondotta al lavoro di Benoît Mandelbrot negli anni ‘80[3], e dagli L-system. Hansmeyer ha fornito un modello di suddivisione, chiamato mesh grammar[4], con elaborati meccanismi di controllo, algoritmi integrati che cercano soluzioni interessanti. Nel 2010, ha esposto un progetto chiamato Subdivided Columns—A New Order costituito da un numero di colonne straordinariamente complesse per il livello di dettaglio offerto dalle suddivisioni[5]. Nel 2013, Benjamin Dillenburger si unì a lui per la progettazione e produzione di --, un’istallazione con un livello finora inedito di dettagli geometrici[6].

1 Stiny, G., & Mitchell, W. J. (1980). “The grammar of paradise: on the generation of Mughul gardens”. Environment and planning B: Planning and design, 7(2), 209-226.
2 Tepavčević, B. & Stojaković, V. (2012). “Shape Grammar in contemporary architectural theory and design”.
3 Mandelbrot, The Fractal Geometry of Nature.
4 Hansmeyer, M and Dillenburger, B (2013). “Mesh Grammars – Procedural Articulation of Form”, Open Systems:
Proceedings of the 18th International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA 2013), 821-829; Hansmeyer, M. (2010). “From Mesh to Ornament”, In Future Cities: ECAADE 2010: Proceedings of the 28th Conference on Education in Computer Aided Architectural Design in Europe, September 15-18, 2010, Zurich, Switzerland, ETH Zurich, Vol. 28, vdf Hochschulverlag AG, 285-293.
5 Michael Hansmayer, “Projects: Columns: Info”, http://www.michaelhansmeyer.com/projects/columns_info.html. Visitato il: 08/09/2020.
6 Digital Grotesque, “Concept,” https://digital-grotesque.com. Visitato il: 08/09/2020.

Figura 4.1_2 (Pagina seguente)
Digital Grotesque
(Fonte: http://www.michael-hansmeyer.com/digital-grotesque-II).

Shape grammar per software di modellazione

La prima applicazione di modellazione 3D commerciale basata su un linguaggio shape grammar, CityEngine, è stata rilasciata dalla società svizzera Procedural Inc. nel 2008.

CityEngine è stato sviluppato all'ETH di Zurigo da Pascal Müller durante la sua ricerca di dottorato presso l'ETH Computer Vision Lab. Il programma utilizza un approccio di modellazione procedurale, il che significa che genera automaticamente modelli urbani attraverso un insieme di regole che perfezionano iterativamente un progetto creando sempre più dettagli[1].

Il programma utilizza una shape grammar, CGA shape, che verrà descritta nella scheda 4. La sua applicazione consente la modellazione di intere città[2], di edifici con un'elevata qualità visiva e livello di dettaglio[3], nonché la modellazione automatica a partire da singole immagini di facciate di risoluzione arbitraria.[4] Nonostante la modellazione procedurale con CGA shape sia stata in origine pensata per la computer grafica, è già stata ampiamente utilizzata anche nella pianificazione urbana, architettura, archeologia e patrimonio culturale digitale. È stata infatti utilizzata per la ricostruzione 3D di Roma[5] e Pompei[6], nonché per la pianificazione delle città sostenibili Masdar e Nanjing e per la ricostruzione procedurale di un edificio in stile Puuc a Xkipché, in Messico[7].

1 Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). "Procedural modeling of buildings". In ACM SIGGRAPH 2006 Papers, 614-623.

2 Müller, P. & Parish, Y. (2001). "Procedural modelling of cities". Proceedings of the 28th annual conference on Computer Graphics and Interactive Techniques – SIGGRAPH '01. New York, USA. New York: ACM Press; 301–308.

3 Müller, P., Zeng, G., Wonka, P., & Van Gool, L. (2007). "Image-based procedural modeling of facades". ACM Trans. Graph., 26(3), 85.

4 Müller, P., Vereenooghe, T., Wonka, P., Paap, I., & Van Gool, L. (2006)."Procedural 3D Reconstruction of Puuc Buildings in Xkipché", The 7th International Symposium on Virtual Reality, Archeology and Cultura Heritage VAST 2006, Eurographics Symposium Proceedings, Aire-la-Ville, 139-146.

5 Dylla, K., Frischer, B., Müller, P., Ulmer, A., & Haegler, S. (2008). "Rome reborn 2.0: A case study of virtual city reconstruction using procedural modeling techniques". Computer Graphics World, 16(6), 62-66.

6 Tepavčević, B. & Stojaković, V. (2012). "Shape Grammar in contemporary architectural theory and design".

7 Ibid.; Haegler, S., Müller, P., & Van Gool, L. (2009). Procedural modeling for digital cultural heritage. EURASIP Journal on Image and Video Processing, 2009(1), 852392.

Scheda 3: Malagueira Grammar

La tesi di dottorato condotta da José P. Duarte presso il MIT Department of Architecture[1], rappresenta una delle migliori applicazioni di shape grammar per la generazioni di soluzioni progettuali originali a partire da linguaggi esistenti. L'obiettivo generale della ricerca è lo sviluppo di un sistema informatico interattivo per la progettazione di alloggi di massa. In particolare, viene sviluppata una shape grammar per sistematizzare le regole di progettazione del sistema insediativo progettato dall'architetto Àlvaro Siza a Malagueira, vicino a Évora, in Portogallo, col fine di generare nuovi alloggi per l'espansione dell'insediamento stesso. La grammatica segue le orme degli studi analitici precedentemente menzionati, pur non essendo volta soltanto a descrivere una famiglia di progetti ma anche a generarne di nuovi.

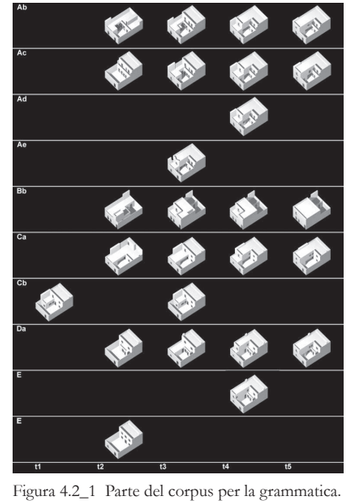


Figura 4.2_1 Parte del corpus per la grammatica.

Corpus

La grammatica si basa su un corpus di trentacinque abitazioni progettate da Siza e i suoi collaboratori tra il 1977 e il 1996. Duarte classifica le abitazioni in due famiglie, a seconda che il cortile sia posizionato davanti (frontyard) o sul retro della casa (backyard). La Figura 4.2_1 mostra le piante delle quattro varianti delle prime case di entrambi i tipi progettate da Siza. La Figura 4.2_2 mostra i diversi sottotipi, Ab, Ac, Bb, Ca, Cb, Da ed E, di una selezione di case nel corpus. Le lettere maiuscole si riferiscono alle zone funzionali (zona giorno, zona notte, servizi, cortile e circolazione), mentre le lettere minuscole corrispondono a specifiche disposizioni spaziali all'interno delle zone funzionali. La lettera "t" seguita da un numero identifica il numero di camere da letto.

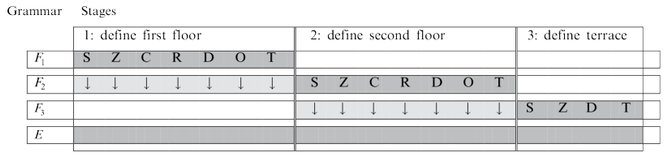

Figura 4.2_2 Piante, sezioni e prospetti di quattro varianti di una casa con cortile sul retro e una casa con cortile sul fronte.

Grammatica

La pianta del primo piano guida la generazione di disegni nella grammatica. La disposizione dei piani superiori è, in larga misura, vincolata a quella del primo piano per motivi strutturali. A sua volta, anche il prospetto è determinato dalle piante. Questa dipendenza è codificata nella grammatica attraverso l'uso di grammatiche sequenziali parallele, una per ciascun piano e un'altra per il prospetto frontale, come mostrato nella figura 4.2_3.

1 Duarte, J. P. (2001). *Customizing mass housing: a discursive grammar for Siza's Malagueira houses* (Doctoral dissertation, Massachusetts Institute of Technology). Lo studio è stato successivamente pubblicato in: Duarte, J.P. (2005). "Towards the mass customization of housing: the grammar of Siza's houses at Malagueira", Environment and Planning B: Planning and Design 32(3), 347-380, da cui sono tratti i materiali trattati in questa scheda.

Figura 4.2_3 Uso di grammatiche sequenziali parallele nella derivazione di una casa di Malagueira.



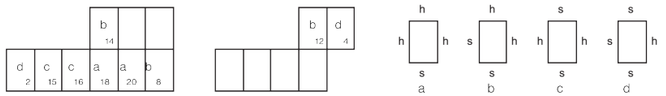
Key: F_1—first floor; F_2—second floor; F_3—terrace; E—elevation; S—start; Z—locate functional zones; C—locate circulation scheme; R—divide zones into rooms; D—introduce details; O—introduce openings; T—terminate.

La derivazione di un progetto nella grammatica passa attraverso tre fasi successive: definizione del primo piano (F1), definizione del secondo piano (F2) e definizione del terrazzo (F3). Man mano che la generazione del primo piano procede, le etichette vengono posizionate sul secondo piano e sul prospetto (E). Al termine della generazione del primo piano, l'etichetta cambia stato, attivando in tal modo la generazione del secondo piano, che procede utilizzando le etichette precedentemente posizionate come riferimento. L'articolazione tra la generazione del secondo piano e la terrazza funziona in modo simile. Ognuna di queste fasi, a sua volta, comprende diversi passaggi. Ad esempio, per il primo piano: individuazione delle zone funzionali, individuazione della scala, divisione delle zone funzionali (stanze), introduzione di dettagli (ad esempio caminetti) e di aperture.

Contesto e forma iniziale

Nella grammatica di Malagueira, la forma iniziale è un rettangolo con un'etichetta "Lot" che rappresenta il lotto di 8 m x 12 m. I lotti sono raggruppati insieme per formare blocchi abitativi. Nella maggior parte dei casi questi blocchi sono rettangolari, ma potrebbero assumere altre forme per adattarsi alla forma delle strade.

Figura 4.2_4 Contesto urbano dei lotti della casa di Malagueira: a) strada solo nella parte anteriore, lotto tipico (simmetria longitudinale); b) strada davanti e di lato (nessuna simmetria); c) strada davanti e dietro (simmetria longitudinale e trasversale); d) strada davanti, dietro e lateralmente (simmetria trasversale).

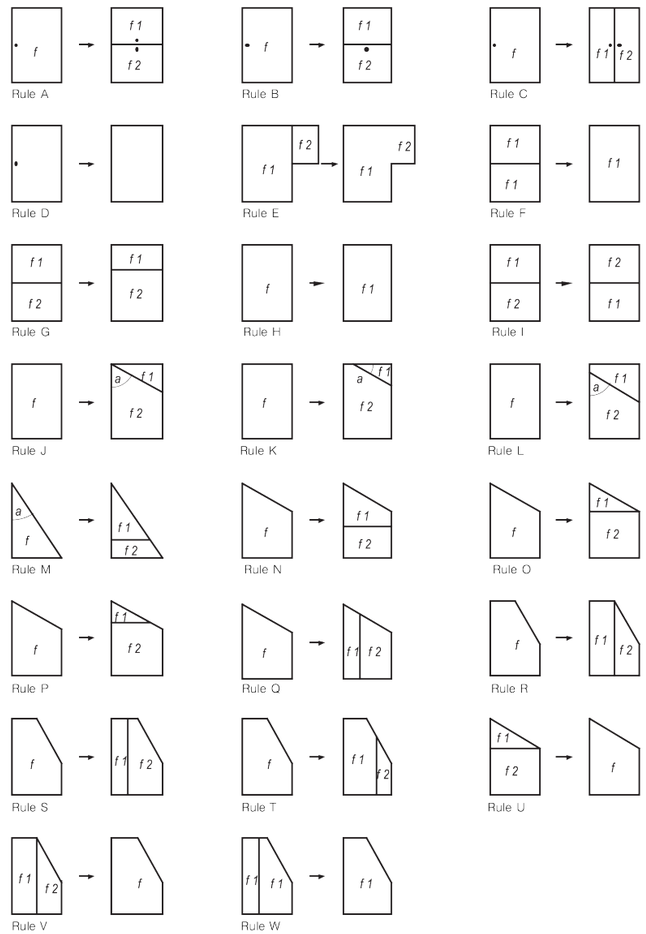


Composizione

In breve, i principi compositivi alla base della generazione di un disegno di Malagueira si basano sulla manipolazione di rettangoli, che rappresentano stanze, mediante regole per dissezionarli, collegarli ed estenderli, nonché regole per assegnare e modificare le funzioni ad essi associate. Nella figura 4.2_5 è mostrato un insieme

semplificato di regole di forma. I passaggi coinvolti nella definizione dell'organizzazione funzionale del primo piano sono invece mostrati nella figura 4.2_6 che illustra come l'applicazione di regole per allocare le zone funzionali generi i cinque schemi di base per le case. Mostra anche come i diversi tipi nel corpus derivino da questi schemi in base alle diverse applicazioni della regola per individuare la scala. Infine, mostra che i sottotipi differiscono l'uno dall'altro in piccole variazioni del layout causate da diverse applicazioni delle regole per la divisione degli ambienti. La Figura 4.2_7 mostra le derivazioni del primo piano di una casa con cortile sul davanti (sottotipo Ab) e una casa con cortile sul retro (sottotipo Bb) con le regole sopra descritte.

Figura 4.2_5 Regole grammaticali.



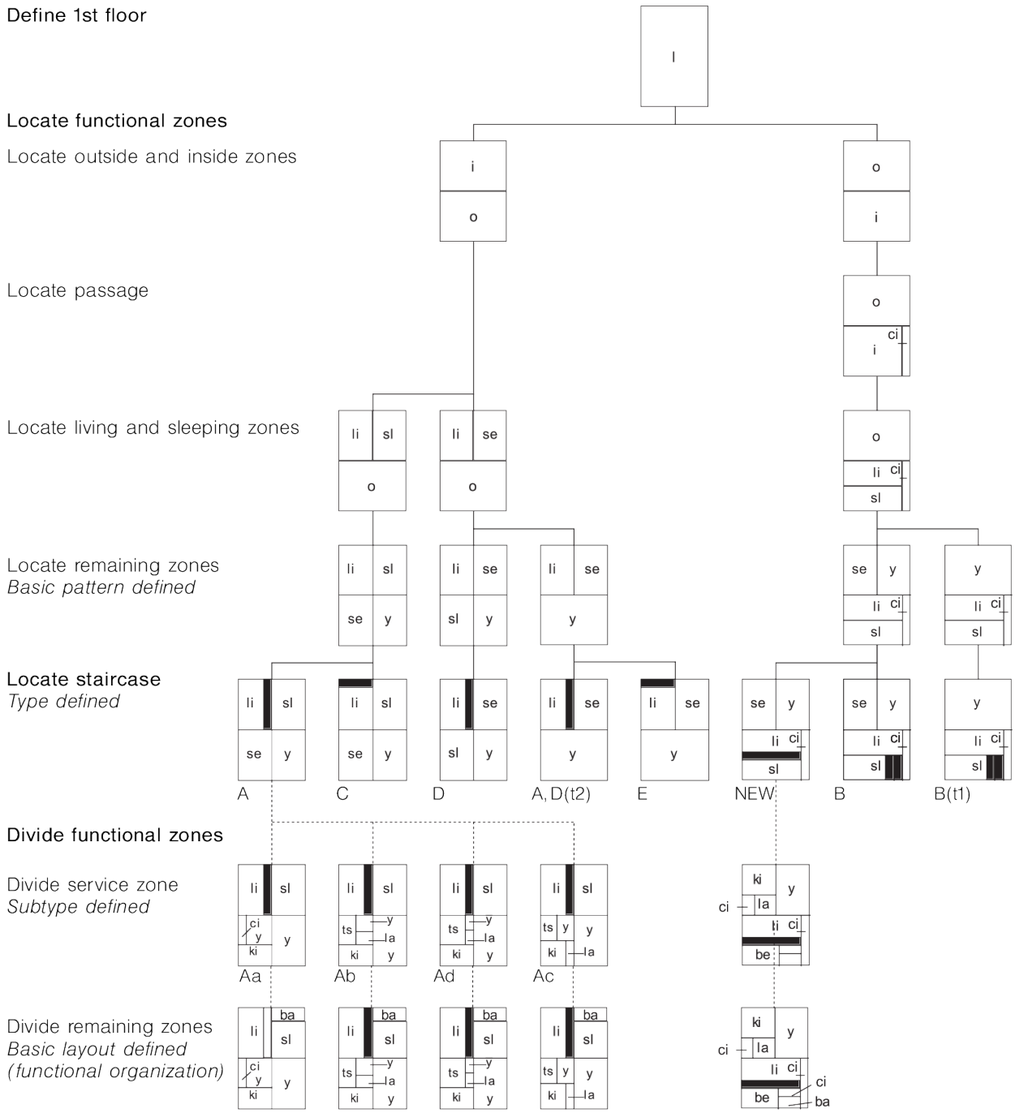


Figura 4.2_6 Diagramma ad albero parziale che mostra la derivazione di pattern, tipi, sottotipi e layout di base. I modelli non sono dimensionati per sottolineare i punti in comune tra i tipi. Il diagramma include i disegni nel corpus e un nuovo disegno. Per chiarezza grafica, l'etichetta ci (circolazione) è stata sostituita da un'area nera campita che indica la scala.

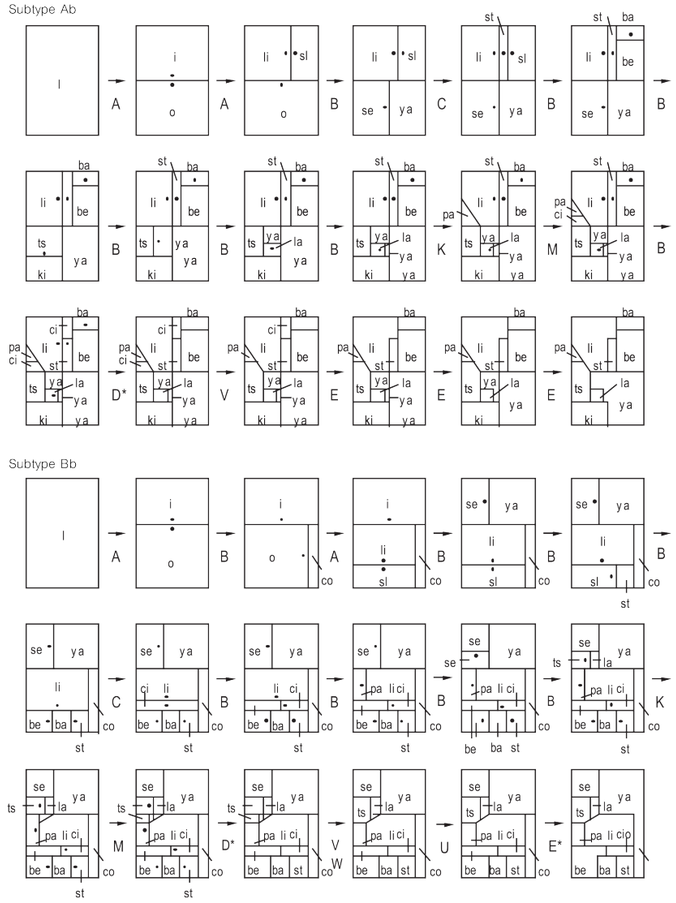


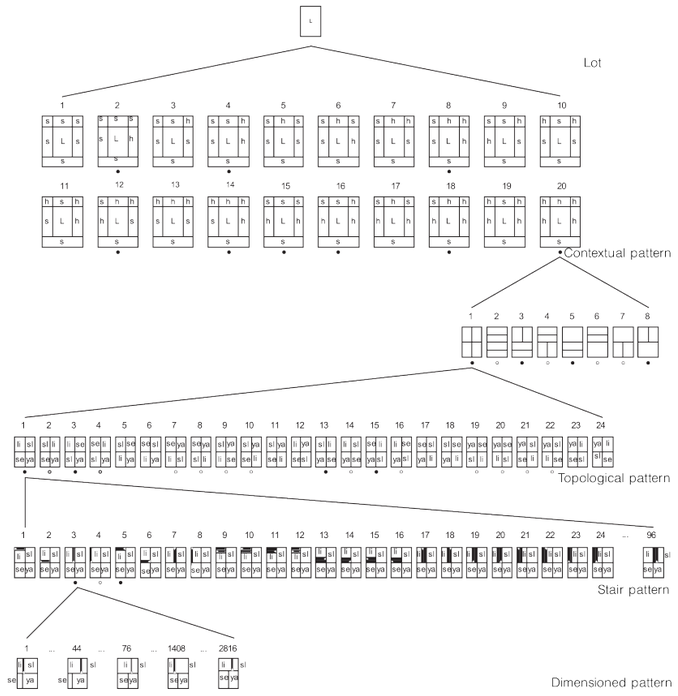
Figura 4.2_7 Derivazioni dell'organizzazione funzionale al primo piano dei sottotipi Ab e Bb. Il simbolo * che segue una lettera della regola (A, B, ecc.) indica che la regola è stata applicata più volte.

Universo di soluzioni e derivazione di progetti originali

Per assicurare che la grammatica, da un lato, generasse un set di soluzioni progettuali il più ampio possibile ma che, dall'altro, queste fossero coerenti con lo stile di Malagueira, si è svolta in tre fasi distinte. Il primo passo è stato sviluppare un insieme esaustivo di regole derivate dai principi compositivi di dissezione e concatenamento delle forme iniziali. Il secondo passo consisteva nel limitare tale insieme di regole ogni volta che appariva non coerente con le regole compositive di Siza. Infine, generare nuovi progetti con un insieme più aperto di regole e quindi chiedere a Siza in persona se li considerava coerenti con la propria "grammatica".

L'universo delle soluzioni progettuali è illustrato nella figura 4.2_8, in una forma semplificata rispetto a quella riportata nella tesi per facilitarne la lettura. Le soluzioni contrassegnate con il simbolo • corrispondono alle case progettate da Siza, mentre quelle contrassegnate con il simbolo ◦ sono considerate "nella grammatica" seguendo un'interpretazione più aperta delle regole di Siza.

Figura 4.2_8 Il potenziale universo di soluzioni della grammatica di Malagueira.



Duarte ha infine condotto diversi esperimenti per testare la grammatica.
Il primo era un tentativo di generare tutti i disegni nel corpus, a partire dal tipo Ab, il primo progetto maturo di Siza a Malagueira. Ogni progetto aggiuntivo ha richiesto la messa a punto della grammatica, risultante nella versione che abbiamo analizzato in questa scheda.
Il secondo esperimento fu quello di derivare dalla grammatica una casa progettata da Siza, ma non presente nel corpus originale. Con l'eccezione dei dettagli per adattare il patio a una forma del lotto leggermente diversa, trapezoidale anziché rettangolare, la grammatica spiegava la derivazione del progetto.
Il terzo esperimento è stato quello di generare nuova abitazione con cortile sul retro, mostrato nella figura 4.2_9, in cui la scala è collocata in una posizione diversa rispetto alle case del corpus (cfr. figura 4.2_6). Le figure 4.2_10 e 4.2_11 mostrano rispettivamente le piante e il modello tridimensionale del nuovo progetto.

" When the new design was shown to Siza amidst other Malagueira designs, he did not notice that it was not his own design. At some point, he seemed confused because he did not remember such a placement of the staircase. But then, he acknowledged its validity and agreed that the design was in the style. When he was told that it was not his design, he was surprised. Then, after a careful analysis of the design and an explanation of its derivation, he acknowledged that the grammar seemed to capture the style. In these three experiments the grammar passed each of the three tests referred to above. Additional experiments were carried out with designs by other authors using the grammar, and Siza maintained his positive opinion. In some cases he mentioned that he would have not designed them for idiosyncratic reasons but he considered them to be in the style"[1].

Figura 4.2_9 Derivazione di una variazione t5 di una nuova abitazione con cortile sul retro. Viene mostrata solo la derivazione della pianta e il prospetto principale del primo piano.

1 J.P. Duarte, "Towards the mass customization of housing: the grammar of Siza's houses at Malagueira", 376-377. Traduzione: " Quando il nuovo progetto è stato mostrato a Siza in mezzo ad altri progetti di Malagueira, non si è accorto che non era il suo design. Ad un certo punto, sembrava confuso perché non ricordava una simile posizione della scala. Ma poi, ha riconosciuto la sua validità e ha convenuto che il progetto era in stile. Quando gli è stato detto che non era il suo progetto, è rimasto sorpreso. Quindi, dopo un'attenta analisi del design e una spiegazione della sua derivazione, ha riconosciuto che la grammatica sembrava catturare lo stile. In questi tre esperimenti la grammatica ha superato ciascuno dei tre test di cui sopra. Ulteriori esperimenti furono condotti con progetti di altri autori usando la grammatica e Siza mantenne la sua opinione positiva. In alcuni casi ha detto che non li avrebbe progettati per ragioni idiosincratiche, ma li ha considerati in stile"..

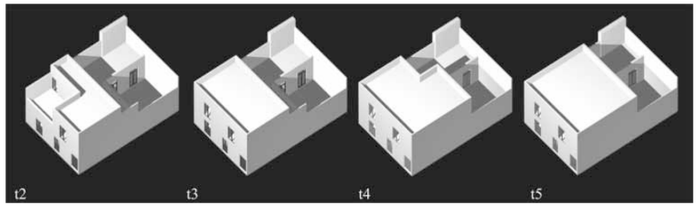


Figura 4.2_10. Piante, sezioni e prospetti di quattro varianti della nuova abitazione.

Figura 4.2_11. Modelli tridimensionali delle quattro varianti.

Scheda 4: CGA Shape

Una delle implementazioni di maggior successo di shape grammar è rappresentata dal lavoro di Müller, Wonka, Haegler, Ulmer e Van Gool che, nel 2006, hanno creato una grammatica sequenziale per la modellazione di edifici in computer grafica, chiamata CGA Shape[1]. Negli anni seguenti, CGA si è evoluta nel più potente "generatore di città" procedurale mai realizzato, chiamato CityEngine, col quale è possibile "far crescere" un modello completamente dettagliato di una città di un milione di abitanti in pochi secondi[2].

La grammatica si ispira ai L-systems[3], tuttavia, a differenza di questi CGA Shape è una grammatica sequenziale simile alle grammatiche di Chomsky, più adatta, secondo gli autori, a gestire la modellazione di edifici:

"While parallel grammars like L-systems are suited to capture growth over time, a sequential application of rules allows for the characterisation of structure i.e. the spatial distribution of features and components. Therefore, CGA Shape is a sequential grammar (similar to Chomsky grammars)[4].

(…) We also use a large set of shape rules not existing in L-systems. Furthermore, the rules governing a biological system do not directly relate to the modelling of buildings. We found that a direct application of L-Systems to architecture overemphasises the idea of growth, a concept that is often counterproductive for the procedural modeling of buildings"[5].

L'idea di modellare contesti urbani usando shape grammar era stata precedentemente esplorata da Parish e Müller[6], che avevano mostrato come generare grandi modelli urbani in cui ogni edificio è costituito da semplici modelli di massa, e da Wonka et al.[7], che avevano dimostrato come generare dettagli geometrici sulle facciate di singoli edifici. CGA Shape nasce dalla combinazione di questi due studi.

Le regole di produzione della grammatica fanno evolvere iterativamente il progetto

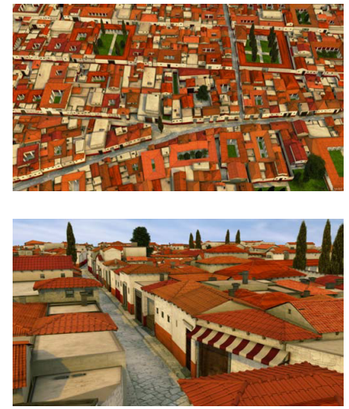


Figura S4_1 Viste del modello procedurale di Pompei realizzato con CityEngine .

1 Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). "Procedural modeling of buildings". In ACM SIGGRAPH 2006 Papers, 614-623. I materiali di questa scheda sono tratti da quest'articolo.

2 Esri, "Esri CityEngine," http://www.esri.com/software/cityengine. (visitato il 31/08/2020).

3 Prusinkiewicz, P. & Lindenmayer, A. (1991). The Algorithmic Beauty of Plants, Springer Verlag.

4 Müller et al.(2006). "Procedural modeling of buildings", 614-615.

5 Müller et al.(2006). "Procedural modeling of buildings", 614-622.

6 Parish, Y. I. H. & Müller, P. (2001). "Procedural modeling of cities", Proceedings of ACM SIGGRAPH 2001, ACM Press, E. Fiume, Ed., 301–308.

7 Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. (2003). "Instant architecture", ACM Transactions on Graphics 22, 3, 669–677.

Figura S4_2 Sinistra: nei metodi di modellazione esistenti, diverse intersezioni indesiderate tagliano le finestre (o altri elementi) in modo innaturale, poiché i volumi non sono consapevoli l'uno dell'altro. A destra: CGA consente la risoluzione di questi conflitti. Inoltre, possiamo posizionare la geometria su poligoni di diverso orientamento come le superfici del tetto. Questo esempio è stato creato usando solo sei regole.

definendo progressivamente i dettagli: si parte da un modello volumetrico grezzo di un edificio, chiamato modello di massa; quindi si definiscono le strutture delle facciate; infine, si aggiungono dettagli come finestre, porte e ornamenti. Le regole, "sensibili al contesto", assicurano che il modello sia coerente, cioè, ad esempio, che entità come finestre o porte non si intersechino con altre pareti, che le porte si aprano su terrazze o sul livello della strada, che le terrazze siano delimitate da ringhiere, ecc. (figura S4_2).

Grammatica

La grammatica di CGA lavora con configurazioni di forme, ossia un insieme finito di forme di base. Una forma è costituita da un simbolo (stringa), una geometria (attributi geometrici) e attributi numerici. Le forme sono identificate dai loro simboli che sono o un simbolo terminale Є Σ, o un simbolo non terminale Є V. Le forme corrispondenti sono chiamate forme terminali e forme non terminali. Gli attributi geometrici più importanti sono la posizione P, tre vettori ortogonali X, Y e Z, che descrivono un sistema di coordinate e un vettore di dimensioni S. Questi attributi definiscono un bounding box orientato nello spazio chiamato scope (figura S4_3).

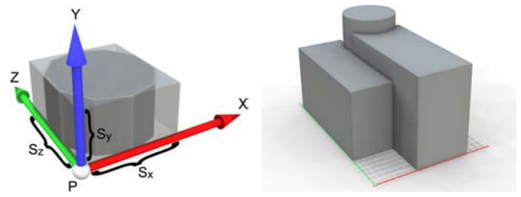


Figura S4_3 Sinistra: lo scope di una forma. Il punto P, insieme ai tre assi X, Y e Z e una dimensione S definiscono un box nello spazio che contiene la forma. A destra: un semplice modello di massa composto da tre primitive di forma.

Processo di produzione

Le regole di produzione sono definite nella seguente forma:

id: predecessor : cond → successor : prob

dove id è un identificatore univoco per la regola, il *predecessor* Є V è un simbolo che identifica una forma che deve essere sostituita con il *successor* e *cond* è una espressione logica che deve essere valutata come vera per poter applicare la regola. La regola è selezionata con probabilità *prob*.

Ad esempio, la regola

1: fac(h) : h > 9 → floor(h/3) floor(h/3) floor(h/3)

sostituisce la forma f ac con tre forme floor, se il parametro h è maggiore di 9.

Le regole generali per modificare le forme sono: T (tx, ty, tz), un vettore di traslazione che viene aggiunto alla posizione dello scope P; Rx (angolo), Ry (angolo) e Rz (angolo), che ruotano il rispettivo asse del sistema di coordinate; e S (sx, sy, sz) imposta le dimensioni dello scope. Si usano le parentesi "[…]" per inserire lo scope corrente in uno stack ("pila"). Qualsiasi simbolo non terminale Є V nella regola verrà creato con lo scope corrente.

Allo stesso modo, il comando I(ob jId) aggiunge un'istanza di una primitiva geometrica con identificatore ob jId. Gli oggetti tipici includono un cubo, un parallelepipedo e un cilindro, ma è possibile utilizzare qualsiasi modello tridimensionale. L'esempio seguente illustra il progetto del modello di massa rappresentato nella figura S4_3 a destra:

1: A→[T(0,0,6) S(8,10,18) I("cube")]
T(6,0,0) S(7,13,18) I("cube") T(0,0,16) S(8,15,8) I("cylinder")

La regola di suddivisione (split rule) divide lo scope corrente lungo un asse. Ad esempio, si consideri la regola per dividere la facciata della figura 4 a sinistra:

1: fac → Subdiv ("Y",3.5,0.3,3,3,3) { floor | ledge | floor | floor | floor}

Il primo parametro descrive l'asse di divisione ("X", "Y" o "Z") e i parametri rimanenti descrivono le dimensioni delle divisioni. Tra le parentesi {…} viene fornito un elenco di forme, separate dal simbolo |. Possono essere usate anche regole di divisione simili per dividere lungo più assi ("XY", "XZ", "YZ" o "XYZ"), divisioni nidificate o combinazioni di divisioni nidificate e regole del L-system.

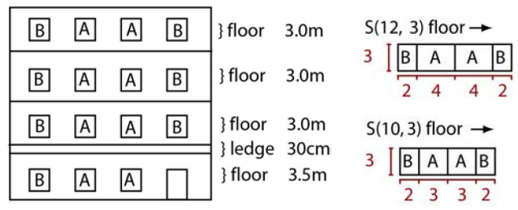


Figura S4_4: A sinistra: un progetto di facciata. A destra: una semplice suddivisione che potrebbe essere utilizzata per gli ultimi tre piani.

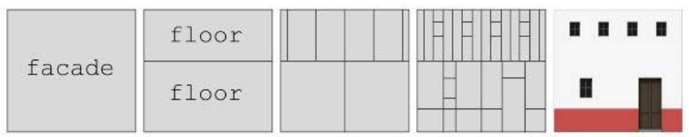


Figura S4_5: Primi 4 passaggi di una sequenza di derivazione della grammatica. A destra il risultato della derivazione.
(Fonte: Van Gool et al. (2007). "3d challenges and a non-in-depth overview of recent progress.").

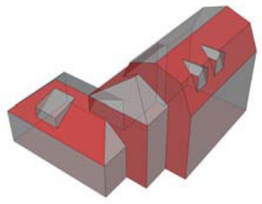


Figura S4_6 L'unione di forme volumetriche semplici porta a poligoni complessi sull'involucro dell'edificio: i poligoni risultanti possono essere concavi, avere molti vertici e più fori (contrassegnati in rosso a sinistra).

Modelli di massa

La grammatica sopra descritta è abbastanza potente da specificare forme complesse. I modelli di massa possono essere creati in due modi: (1) Ci viene dato un lotto edificabile come assioma della grammatica. Quindi siamo in grado di generare modelli di massa, utilizzando varie trasformazioni, facendo attenzione che la massa dell'edificio non sporga dai confini del lotto. (2) Si importano dati da un database GIS o si importa un modello architettonico esistente, in questo caso si classificano i modelli di massa importati come forme di base già definite nel nostro vocabolario di forme. Se ciò non è possibile, si utilizza un'impronta dell'edificio estrusa a cui si associa un tetto generico come primitive.

La strategia di modellazione consiste nel partire utilizzando scope tridimensionali per posizionare i volumi che formano il modello di massa. Quindi si generano scope bidimensionali allineati con le superfici delle facciate e dei tetti estraendo le facce delle forme tridimensionali. Gli scope bidimensionali risultanti saranno correttamente allineati e parametrizzati. Allo stesso modo possiamo estrarre i bordi generando scope unidimensionali. La grammatica può quindi procedere a perfezionare le divisioni in quadrati e triangoli (e in casi molto limitati poligoni generali). È anche importante notare che dopo che una forma (scope) è stata ridotta a due dimensioni, viene spesso sostituita con una tridimensionale dalle successive applicazioni di regole.

Per garantire che il design sia coerente bisogna:

1) testare la sovrapposizione spaziale (occlusione), ossia verificare le intersezioni tra le forme. Questo consente di evitare di collocare elementi di facciata come finestre e porte sull'intersezione di forme volumetriche che costituiscono il modello di massa (vedi figura S4_2);

2) testare linee e piani vicini nella configurazione della forma (linee di snap). Questo permette di migliorare ulteriormente il layout della struttura della facciata, attraverso la modifica delle regole di forma esistenti per agganciare una faccia o una linea dominante (forme snap) nella configurazione della forma.

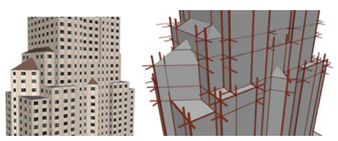


Figura S4_7: A sinistra: un edificio generato con linee snap. Le linee sottili sull'edificio mostrano gli scope delle forme finali che illustrano la struttura della grammatica. Notare come i livelli del pavimento vengono allineati automaticamente su tutti i solidi, ad es. un piano più alto è stato forzato al di sotto della rastremazione (caratteristica comune dei grattacieli). A destra: le linee di snap usate durante la costruzione.

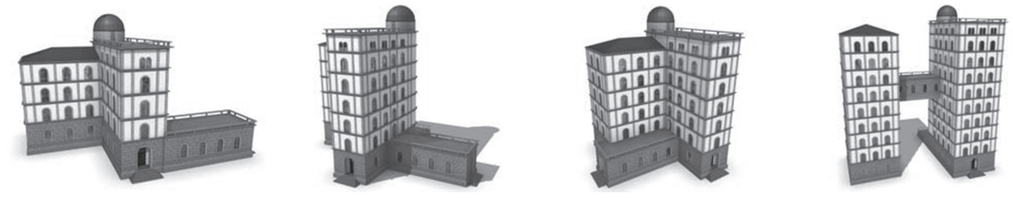


Figura S4_8 Esempio di applicazione di CGA Shape.

CAPITOLO 5

NAIVE GRAMMARS

In questo capitolo vengono analizzati approcci alternativi all'uso di shape grammar in cui le regole generative sono più semplici ("grammatiche ingenue").

Nel primo paragrafo viene chiarito perchè questo genere di grammatiche necessitano di essere integrate con altri metodi per il controllo del processo generativo.

Nelle schede verranno analizzate le principali tecniche utilizzate per guidare il processo generativo: ricottura simulata (sheda 5); algoritmi evolutivi (scheda 6); reti neurali profonde (scheda 7); apprendimento per rinforzo (scheda 8).

5.1_Grammatiche ingenue e meccanismi di controllo

Come abbiamo visto nel primo capitolo, le shape grammars, nella loro forma tradizionale, risultano spesso difficili da creare e modificare in quanto richiedono la codifica precisa e ben articolata, all'interno delle loro regole, delle conoscenze e competenze di progettisti esperti.

L'uso di grammatiche più semplici, del resto, ha dei pro e dei contro: sebbene infatti queste possano da un lato portare a una grande quantità e varietà nei risultati, potrebbero dall'altro anche portare alla generazione di molte soluzioni inutili. Ruiz-Montiel et al.[1] definiscono *ingenue* questo tipo di grammatiche perché un'esecuzione arbitraria delle loro regole senza alcun meccanismo guida aggiuntivo non garantirebbe progetti realizzabili. È necessario pertanto combinarle con metodi specifici per guidare il processo di generazione e assicurare l'adempimento dei requisiti di progettazione. In tal modo si eviterebbero le difficoltà legate alle shape grammar *esperte* tradizionali, lasciando la garanzia della fattibilità principalmente nelle mani del meccanismo di controllo[2].

I principali studi sulla generazione di progetti *goal-oriented* con grammatiche ingenue fanno ricorso a modelli di apprendimento automatico[3] per fare in modo che l'algoritmo, invece di raggiungere direttamente le soluzioni, possa "imparare" come generarle: dopo aver impostato la grammatica, vi è dunque una fase di apprendimento che guida l'esecuzione della stessa verso soluzioni fattibili.

I primi lavori di questo tipo hanno generalmente definito la questione in termini di ottimizzazione, con una funzione-obiettivo quantificabile da massimizzare o minimizzare e un insieme opzionale di vincoli di progettazione[4]. Sono state utilizzate due tecniche di ottimizzazione principali: ricottura simulata e algoritmi evolutivi[5]. Più di recente sono state fatte sperimentazioni anche con modelli di apprendimento più direttamente ispirati al funzionamento della mente umana, come le reti neurali artificiali[6].

Di seguito verranno analizzati i vari approcci al problema, mettendo in risalto vantaggi e svantaggi di ciascuno.

---

1 Ruiz-Montiel, M., Boned, J., Gavilanes, J., Jiménez, E., Mandow, L., & PéRez-De-La-Cruz, J. L. (2013). "Design with shape grammars and reinforcement learning". *Advanced Engineering Informatics*, *27*(2), 230-245.
2 Ibid.
3 Cfr. Appendice 1
4 Ibid.
5 Cfr. Appendice 2
6 Cfr. Appendice 1

Scheda 5: S.G. + ricottura simulata (Shape Annealing)

Shape Annealing[1] è un metodo che combina shape grammar con la tecnica di ottimizzazione chiamata "ricottura simulata" (simulated annealing)[2], una tecnica che simula il processo fisico della ricottura dei metalli, ovvero il riscaldamento intenso e quindi il raffreddamento graduale fino a raggiungere uno stato di equilibrio a bassa energia.

Analogamente, in Shape Annealing, si cerca una forma che minimizzi il valore di una data funzione obbiettivo, interpretata come l'energia della forma stessa. Una serie di vincoli determina inoltre le transizioni valide nell'applicazione delle regole della grammatica. Ad ogni passo della derivazione, una regola che porta a una nuova forma fattibile viene selezionata casualmente. Se la nuova forma ottenuta applicando questa regola ha un'energia inferiore a quella corrente, la transizione viene automaticamente accettata. Tuttavia, se la nuova forma ha un'energia maggiore, la transizione viene accettata solo con una probabilità che dipende dalla differenza di energia e dalla temperatura del processo. Inizialmente, quando la temperatura è alta, il processo può facilmente sfuggire all'optima locale, tuttavia man mano che il processo va avanti e la temperatura si riduce, questa possibilità diventa meno probabile. La sequenza stocastica delle applicazioni delle regole continua fino a quando la forma corrente non può essere ulteriormente migliorata o viene raggiunto un limite al numero di derivazioni delle regole[3].

Shape Annealing è stata applicata principalmente a problemi di progettazione strutturale, particolarmente rilevanti sono, ad esempio, le sperimentazioni di Shea e Ca-

1 Cagan, J. & Mitchell, W.J. (1993). "Optimally directed shape generation by shape annealing", Environment and Planning B: Planning and Design 20, 5–12; Shea, K. & Cagan, J. (1997). "Innovative dome design: applying geodesic patterns with shape annealing", Artificial Intelligence for engineering design, analysis and manufacturing 11, 379–394; Shea, K. & Cagan, J. (1999). "Languages and semantics of grammatical discrete structures", Artificial Intelligence for Engineering Design, Analysis and Manufacturing 13, 241–251; Reddy, G. & Cagan, J. (1995). "An Improved Shape Annealing Algorithm For Truss Topology Generation." ASME. J. Mech. Des. June 1995; 117(2A): 315–321.

2 Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.

3 Cagan, J. & Mitchell, W.J. (1993). "Optimally directed shape generation by shape annealing"; Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning".

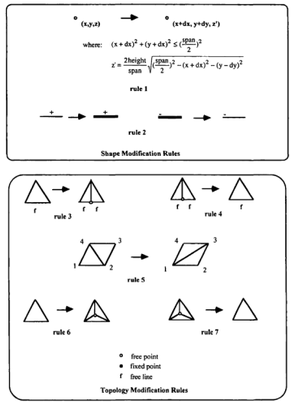


Figura 5.2_1 Shape Annealing: Grammatica utilizzata da Shea & Cagan per progetti di cupole. (Fonte: Shea, K. & Cagan, J. (1997). "Innovative dome design: applying geodesic patterns with shape annealing").

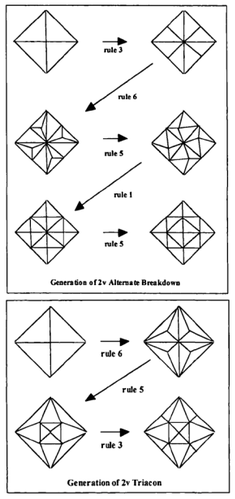


Figura 5.2_2 Shape Annealing: Generazione di moduli standard. (Fonte: Shea, K. & Cagan, J. (1997). "Innovative dome design: applying geodesic patterns with shape annealing").

gan per capriate[1] e cupole geodetiche[2]. Il suo utilizzo può portare a un risultato quasi ottimale se i parametri dell'algoritmo sono configurati correttamente. Oltre all'ottimizzazione dei parametri, il processo consiste sostanzialmente nel calcolare la funzione-obiettivo e verificare se i vincoli vengono violati. Uno svantaggio di questo metodo è che ogni esecuzione dell'algoritmo porta solo a un'unica soluzione: se miriamo a ottenere più soluzioni distinte, dobbiamo eseguire ripetutamente l'algoritmo di ottimizzazione[3].

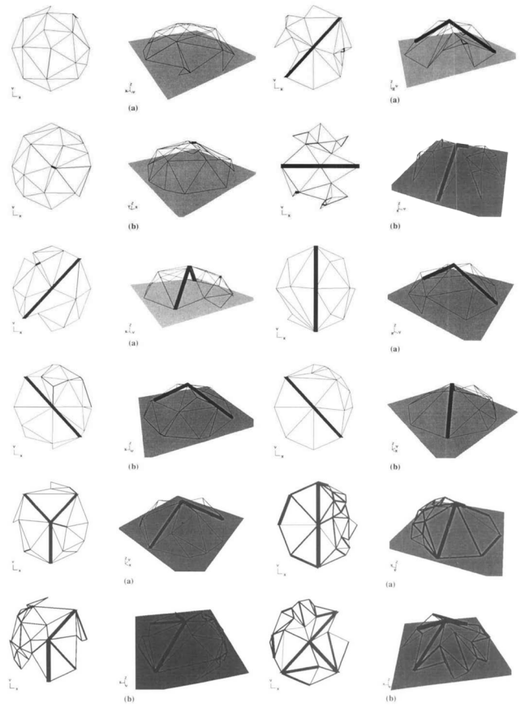


Figura 5.2_3 Shape Annealing: esempi di cupole ottenuti. (Fonte: Shea, K. & Cagan, J. (1997). "Innovative dome design: applying geodesic patterns with shape annealing").

1 Shea, K. & Cagan, J. (1999). "Languages and semantics of grammatical discrete structures"; Shea, K. & Cagan, J. (1999). "The design of novel roof trusses with shape annealing: assessing the ability of acomputational method in aiding structural designers with varying design intent", Design Studies 20, 3–23.

2 Shea, K. & Cagan, J. (1997). "Innovative dome design: applying geodesic patterns with shape annealing".

3 Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning".

Scheda 6: S.G. + algoritmi evolutivi

Gli algoritmi evolutivi, o genetici, sono stati utilizzati in molti campi. Questo approccio si fonda sulla logica della selezione naturale e permette di generare casualmente molte soluzioni e testarle per eliminare quelle chiaramente non adatte. Le soluzioni rimanenti possono mutare e combinarsi, generando nuove soluzioni con caratteristiche "ereditate" dai loro predecessori, ma anche con caratteristiche inedite, grazie all' introduzione casuale di mutazioni[1].
Gero et al., nell'articolo "Evolutionary learning of novel grammars for design improvement"[2], utilizzano il formalismo dell'algoritmo genetico come costrutto computazionale per realizzare un processo di apprendimento, inteso come esplorazione e modifica dello spazio all'interno del quale avvengono la ricerca e il processo decisionale:

"[Genetic algorithms] can also be considered a search process, searching for better individuals in the space of all possible individuals. In practice, individuals represent points in a state space, while the environment provides a measure of "fitness" that helps identify better individuals. Genetic algorithms are a robust, parallel search process requiring little information to search effectively. As such they are well suited to the task of exploring and learning about large and complex design spaces"[3].

Nell'articolo è dunque presentato un processo, esemplificato utilizzando shape grammar per realizzare una sezione di trave, in cui viene appresa una nuova grammatica che produce un nuovo spazio di stato per il problema:

"The class of learning we are presenting here restructures the design knowledge so as to produce new points in design spaces which lie outside the realm of the state spaces defined by the given examples. These points are discovered during an

1 Kalay, Architecture's New Media, 202-203; Gero, John S. (1996). "Creativity, emergence and evolution in design", Knowledge-Based Systems 9: 435-448. Vedi Appendice 2.

2 Gero, J. S., Louis, S. J., & Kundu, S. (1994). "Evolutionary learning of novel grammars for design improvement", Artificial Intelligence for Engineering Design, Analysis and Manufacturing 8, 83–94.

3 Ibid. Traduzione: "[Gli algoritmi genetici] possono anche essere considerati un processo di ricerca, per trovare gli individui migliori nello spazio di tutti gli individui possibili. In pratica, gli individui rappresentano punti in uno spazio di stato, mentre l'ambiente fornisce una misura di "fitness" che aiuta a identificare individui migliori. Gli algoritmi genetici sono un robusto processo di ricerca parallela che richiede poche informazioni per cercare in modo efficace. In quanto tali, sono adatti al compito di esplorare e conoscere spazi di progettazione ampi e complessi".

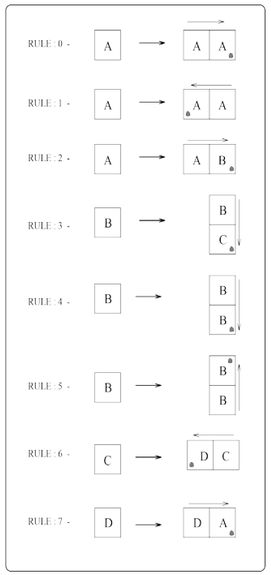


Figura 5.3_1 SG + algoritmi genetici: regole della grammatica di Gero et al. (Fonte: Gero, J. S., Louis, S. J., & Kundu, S. (1994). "Evolutionary learning of novel grammars for design improvement").

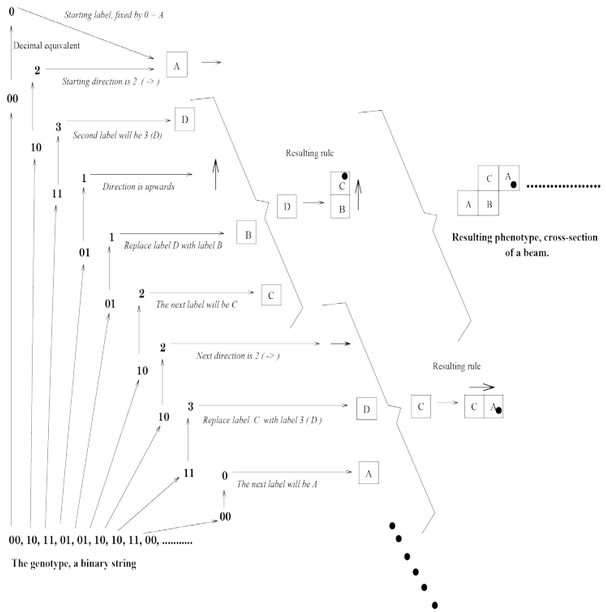


Figura 5.3_3 SG + algoritmi genetici: tipica stringa genotipica e sua struttura risultante (Fonte: Gero, J. S., Louis, S. J., & Kundu, S. (1994). "Evolutionary learning of novel grammars for design improvement").

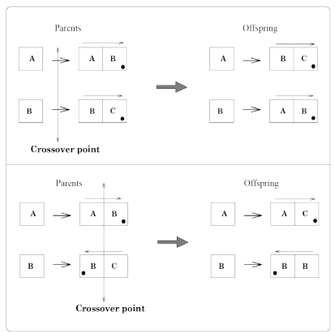


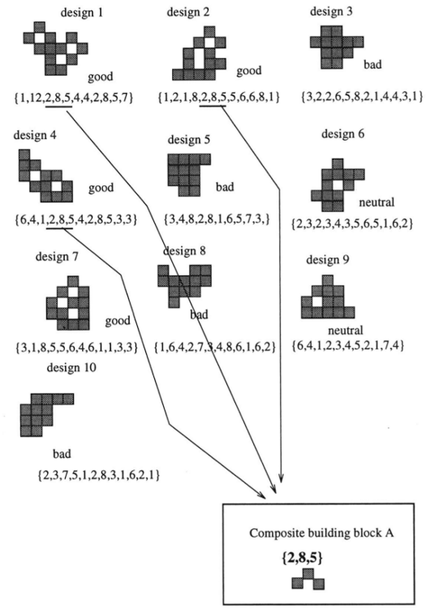
Figura 5.3_2 SG + algoritmi genetici: generazione di nuove regole attraverso il crossover (Fonte: Gero, J. S., Louis, S. J., & Kundu, S. (1994). "Evolutionary learning of novel grammars for design improvement").

evolutionary process by the creation and exploration of possibly novel state spaces using the restructured knowledge. The learning program is not presented with examples, rather it is presented with a set of rules in a grammar which, when executed, produces examples. The state spaces are not defined by the given examples but by the grammar that produces those examples. As a new grammar is learned the state spaces change, perhaps drastically. This type of system can be seen as more of a generative or formation system rather than a classification system. Thus the system tries to achieve better designs by learning to restructure knowledge which is capable of producing novel designs which could not be produced with the original knowledge"[1].

1 Ibid. Traduzione: "La classe di apprendimento che presentiamo qui ristruttura la conoscenza del design in modo da produrre nuovi punti negli spazi di progettazione che si trovano al di fuori del regno degli spazi di stato definiti dagli esempi forniti. Questi punti vengono scoperti durante un processo evolutivo mediante la creazione e l'esplorazione di spazi di stato possibilmente nuovi utilizzando la conoscenza ristrutturata. Il programma di apprendimento non è presentato con esempi, piuttosto è presentato con un insieme di regole in una grammatica che, una volta eseguita, produce esempi. Gli

Sono descritte due diverse applicazioni di algoritmi genetici a shape grammar. Nel primo approccio (routine design), cercano l'ordine di esecuzione (o derivazione) delle regole grammaticali che ottimizza un determinato insieme di vincoli. Questo può essere fatto usando le sequenze di regole come genotipi e valutando l'idoneità dei risultati ottenuti. Nel secondo, più ambizioso, le stesse regole grammaticali sono codificate per essere manipolate dall'algoritmo genetico. In questo modo vengono prodotte nuove grammatiche che potrebbero condurre a progetti migliori o perfino innovativi[1].

Gero e Kazakov[2] nel 1996 propongono uno studio che si ispira alla kindergarten grammar di Stiny.



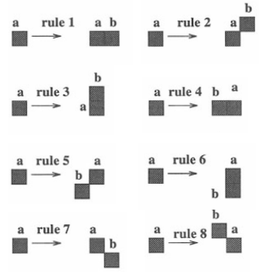


Figura 5.3_4 SG + algoritmi genetici: grammatica utilizzata da Gero e Kazakov (Fonte: Gero, J.S. & Kazakov, V.A. (1996). "Evolving building blocks for design using genetic engineering: a formal approach").

Figura 5.3_5 SG + algoritmi genetici: identificazione delle sub-sequenze di regole che caratterizzano le soluzioni "buone". (Fonte: Gero, J.S. & Kazakov, V.A. (1996). "Evolving building blocks for design using genetic engineering: a formal approach").

spazi di stato non sono definiti dagli esempi forniti ma dalla grammatica che produce quegli esempi. Man mano che si impara una nuova grammatica, gli spazi degli stati cambiano, forse drasticamente. Questo tipo di sistema può essere visto più come un sistema generativo o di formazione piuttosto che come un sistema di classificazione. Pertanto il sistema cerca di ottenere progetti migliori imparando a ristrutturare la conoscenza che è in grado di produrre nuovi progetti che non potrebbero essere prodotti con la conoscenza originale".

1 Ibid.; Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning".

2 Gero, J. S., & Kazakov, V. A. (1996). "Evolving building blocks for design using genetic engineering: a formal approach". In Advances in formal design methods for CAD, Springer, Boston, MA, 31-50.

Una serie iniziale di geni, che rappresentano blocchi da costruzione elementari, si evolve in una serie di geni complessi, che rappresentano blocchi da costruzione "obiettivo" con migliori possibilità di produrre progetti che presentano le caratteristiche desiderate rispetto ai blocchi iniziali.
Gli algoritmi genetici vengono utilizzati per evolvere le regole di forma identificando le sub-sequenze di regole che appaiono negli "individui adatti" delle popolazioni e non in quelli "non adatti". Queste sub-sequenze sono state usate per creare nuove regole complesse che vengono combinate con un algoritmo genetico standard al fine di distribuire gli elementi compositivi secondo determinati criteri.

Approcci simili sono stati utilizzati in molti lavori più recenti: Ang et al.[1] hanno combinato shape grammar e algoritmi genetici per progettare bottiglie di Coca-Cola, mediante regole parametriche e codificando i parametri delle sequenze di regole all'interno del genotipo; un sistema simile ma più complesso è descritto da Lee e Tang[2] che utilizza la programmazione genetica per determinare i parametri delle regole per la progettazione del corpo di una fotocamera; Chouchoulas[3] usa un algoritmo genetico per evolvere sequenze di regole al fine di generare edifici per appartamenti capaci per soddisfare determinati criteri; O'Neill et al.[4] impiegano un algoritmo genetico per progettare coperture.
Secondo Ruiz-Montiel et al.[5] un vantaggio offerto da tutti gli approcci basati sull'evoluzione è l'ingegneria della conoscenza più semplice. La fattibilità dei progetti risultanti è rafforzata da una funzione fitness che li valuta in base a una serie di criteri. Tra gli aspetti negativi menzionano invece la difficoltà di impostazione dei parametri e la scarsa varietà di soluzioni:
"The set of generated solutions is often restricted in the sense that it has a relatively small size compared to the diversity potential of shape grammars. Some works[6] offer just a single solution to a given problem (an optimal or near-optimal one), while others[1] provide few distinct, feasible solutions (less than 10), which are picked from the last generation"[2].

1 Ang, M.C., Chau, H.H., Mckay, A., de Pennington, A.(2006). "Combining evolutionary algorithms and shape grammars to generate branded product design", in: J.S. Gero (Ed.), Design Computing and Cognition '06, Springer, Netherlands, Dordrecht, 521–539.

2 H.C. Lee, M.X.Tang (2009). "Evolving product form designs using parametric shape grammars integrated with genetic programming", Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23, 131–158.

3 O. Chouchoulas (2003). Shape Evolution. An Algorithmic Method for Conceptual Architectural Design Combining Shape Grammars and Genetic Algorithms, Ph.D. Thesis, University of Bath.

4 M. O'Neill, J.McDermott, J.M.Swafford, J.Byrne, E.Hemberg, A.Brabazon, E. Shotton, C.McNally, M.Hemberg (2010). "Evolutionary design using grammatical evolution and shape grammars: designing a shelter", International Journal of
Design Engineering 3.

5 Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning".

6 M.C. Ang, H.H.Chau, A.Mckay, A.de Pennington, "Combining evolutionary algorithms and shape grammars to generate branded product design", in: J.S. Gero (Ed.), Design Computing and Cognition '06, Springer, Netherlands,
Dordrecht, pp.521–539; H.C. Lee, M.X.Tang (2009). "Evolving product form designs using parametric shape grammars integrated with genetic programming", Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23, 131–158.

1 J.S. Gero, S.J.Louis, S.Kundu (1994). "Evolutionary learning of novel grammars for design improvement", Artificial Intelligence for Engineering Design, Analysis and Manufacturing 8, 83–94; J.S. Gero, V.A.Kazakov (1996). "Evolving building blocks for design using genetic engineering: a formal approach", in: Advances in Formal Design Methods for CAD, Hall, pp.31–50; O. Chouchoulas (2003). Shape Evolution. An Algorithmic Method for Conceptual Architectural Design Combining Shape Grammars and Genetic Algorithms, Ph.D. Thesis, University of Bath.

2 Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning", 234. Traduzione: "L'insieme delle soluzioni generate è spesso limitato nel senso che ha una dimensione relativamente piccola rispetto al potenziale di diversità delle shape grammars. Alcuni lavori offrono una sola soluzione a un dato problema (ottimale o quasi ottimale), mentre altri forniscono poche soluzioni distinte e fattibili (meno di 10), che vengono scelte dall'ultima generazione".

Scheda 7: S.G. + deep learning

La modellazione procedurale basata di CGA grammar, che abbiamo affrontato nel capitolo 4, ha richiesto, in molte applicazioni, strumenti di controllo aggiuntivi per soddisfare obiettivi specifici: rendere i loro output simili ad esempi dati[1], adattarsi a una forma-obiettivo[2] o rispettare vincoli funzionali[3].
Alcune sperimentazioni interessanti utilizzano reti neurali profonde[4] per guidare i processi di modellazione.
Nel 2016 Ritchie et al.[5] hanno utilizzato le DNN per ottenere quelli che hanno definito "neurally-guided procedural models" per la generazione di grafica 2D.

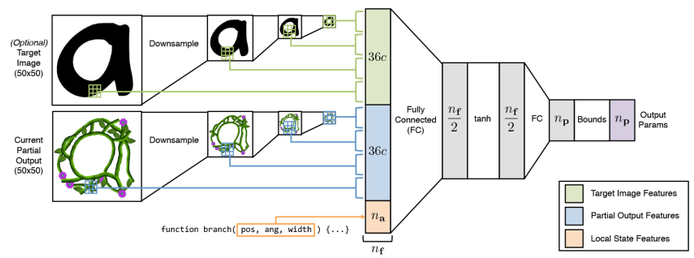


Figura 5.4_1 Architettura delle reti dei neurally-guided procedural models. (Ritchie et al. (2016). "Neurally-guided procedural models: Amortized inference for procedural graphics programs using neural networks").

1 Stava, O., Pirk, S., Kratt, J., Chen, B., Měch, R., Deussen, O., & Benes, B. (2014, September). Inverse procedural modelling of trees. In Computer Graphics Forum, Vol. 33, No. 6, 118-131; Dang, M., Lienhard, S., Ceylan, D., Neubert, B., Wonka, P., & Pauly, M. (2015). Interactive design of probability density functions for shape grammars. ACM Transactions on Graphics (TOG), 34(6), 1-13.
2 Prusinkiewicz, P., James, M., & Měch, R. (1994, July). Synthetic topiary. In Proceedings of the 21st annual conference on Computer graphics and interactive techniques (pp. 351-358); Talton, J. O., Lou, Y., Lesser, S., Duke, J., Měch, R., & Koltun, V. (2011). Metropolis procedural modeling. ACM Transactions on Graphics (TOG), 30(2), 1-14; Ritchie, D., Mildenhall, B., Goodman, N. D., & Hanrahan, P. (2015). Controlling procedural modeling programs with stochastically-ordered sequential monte carlo. ACM Transactions on Graphics (TOG), 34(4), 1-11.
3 Ritchie, D., Lin, S., Goodman, N. D., & Hanrahan, P. (2015, May). Generating design suggestions under tight constraints with gradient-based probabilistic programming. In Computer Graphics Forum, Vol. 34, No. 2, 515-526.
4 Cfr. Appendice 1.
5 Ritchie, D., Thomas, A., Hanrahan, P., & Goodman, N. (2016). "Neurally-guided procedural models: Amortized inference for procedural graphics programs using neural networks". In Advances in neural information processing systems, 622-630.

Nel 2018 Zeng et al.[1] hanno proposto un approccio alla ricostruzione 3D, denominato Neural Procedural Reconstruction (NPR), che combina CGA grammar con reti neurali profonde (DNN) per la ricostruzione di intere strutture di edifici a partire da dati input incompleti. Tali input sono nuvole di punti 3D ottenuti dalla scansione di ambienti urbani ottenuti con tecnologia LiDAR[2].

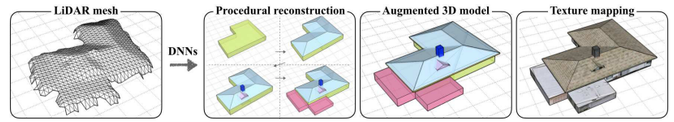


Figura 5.4_2 La ricostruzione procedurale neurale impara ad applicare regole grammaticali per ricostruire modelli CAD a partire da punti 3D. (Zeng et al. (2018). "Neural procedural reconstruction for residential buildings").

La grammatica proposta, ottenuta modificando una grammatica predefinita in CityEngine[3], si compone di sette regole, che formano cinque fasi di ricostruzione (vedi Fig. 5.5_2). Le regole vengono applicate alle fasi in un ordine sequenziale fisso[4]. Ogni regola è associata a più rami e ogni ramo ha i suoi parametri geometrici.

- La "foundation-rule" determina, a partire da una forma 2D, il volume della casa. Sono definite sei tipologie di abitazione, che rappresentano i rami a cui le regole sono associate: I, II, III, L, U o C. I primi tre tipi (I, II e III) determinano abitazioni costituite rispettivamente da uno, due e tre blocchi rettangolari. I tipi L e U si riferiscono, ovviamente, agli edifici a forma di L o a forma di U. L'ultimo tipo, C, è per case con forme complesse non previste dalla grammatica o strutture non architettoniche come gli alberi, dove il sistema fermerebbe il processo di ricostruzione. I parametri geometrici sono la posizione, la forma e l'altezza di ciascun blocco fondamentale.
- Tre "roof-rules" determinano la struttura del tetto per ciascun blocco di base. I parametri geometrici ne determinano forma e altezza.
- La "dormer-rule" aggiunge un numero arbitrario di abbaini ad ogni componente dato dalla prima regola. Questi vengono modellati come volumi a cui viene aggiunto un tetto a due falde. Pertanto, i parametri geometrici sono la posizione e la forma del volume più l'altezza del tetto.
- La "chimney-rule" funziona in maniera analoga, tranne per il fatto che il tetto

1 Zeng, H., Wu, J., & Furukawa, Y. (2018). "Neural procedural reconstruction for residential buildings". In Proceedings of the European Conference on Computer Vision (ECCV), 759-775.
2 "LIDAR (acronimo dall'inglese Light Detection and Ranging o Laser Imaging Detection and Ranging) è una tecnica di telerilevamento che permette di determinare la distanza di un oggetto o di una superficie utilizzando un impulso laser". https://it.wikipedia.org/wiki/Lidar (visitato il: 16/09/2020).
3 EsriCityEngine: http://www.esri.com/software/cityengine (2017)
4 Zeng, H. et al. (2018). "Neural procedural reconstruction for residential buildings", 761.

è piano e la forma quadrata.

- La "garage-rule" aggiunge un numero arbitrario di sottostrutture circostanti come garage, o balconi. Il tetto è piano.

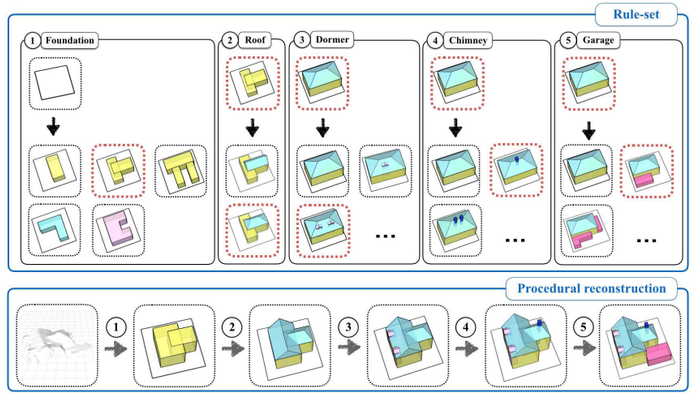


Figura 5.4_3 Il sistema NPR per edifici residenziali ha 5 fasi, costituite da sette regole shape grammar. L'ordine di applicazione delle regole viene fissato mentre le DNN vengono addestrate a 1) selezionare un ramo di regole e 2) regredire i parametri geometrici per ciascuna applicazione di regole. (Zeng et al. (2018). "Neural procedural reconstruction for residential buildings").

La ricostruzione procedurale neurale (NPR) applica le regole in un ordine sequenziale fisso, risolvendo due compiti fondamentali ad ogni applicazione della regola: 1) classificare un ramo della regola e 2) regredire i parametri geometrici associati al ramo.

L'immagine di input utilizzata per la DNN ha 4 canali che codificano la normale alla superficie e la profondità, i cui valori sono normalizzati nella gamma di intensità [0, 255]. In fase di preelaborazione, viene applicata la DNN di rettifica dell'orientamento introdotta da Mousavian et al.[1] e si assume che le strutture degli edifici siano allineate agli assi.

Si inizia con la Foundation-rule. Una variante di ResNet[2] esegue la classificazione del ramo della regola. Si tratta di una rete neurale molto profonda, composta da 152 layer, che con l'ausilio di una tecnica nota come skip connection può calcolare il gradiente in modo più veloce ed efficace di quanto non permetta la backpropagation[3]. Infatti, invece di aspettare che il gradiente si propaghi indietro un livello alla volta, la skip connection consente di raggiungere i nodi iniziali efficacemente

1 Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J. (2017). "3d bounding box estimation using deep learning and geometry", IEEE Conference on Computer Vision and Pattern Recognition.

2 He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep residual learning for image recognition". In Proceedings of the IEEE conference on computer vision and pattern recognition, 770-778.

3 Cfr. Appendice 1.

saltando quelli intermedi[1].

La variante utilizzata nella sperimentazione aggiunge un ulteriore livello completamente connesso tra il pool globale e i livelli di output con SoftMax[2].

Non essendo possibile la regressione diretta dei parametri, per la regressione viene utilizzata una rete autoencoder (Figura 5.4_4), che codifica un input in una rappresentazione compressa (attraverso l'encoder) e restituisce, a partire da essa, una ricostruzione dell'input (attraverso il decoder)[3].

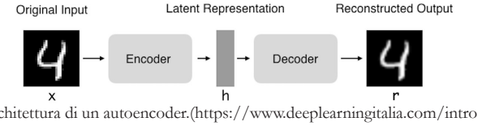


Figura 5.4_4 Architettura di un autoencoder.(https://www.deeplearningitalia.com/introduzione-agli-autoencoder)

In tal modo vengono rilevati i "possibili candidati" dei vertici dei poligoni, che rappresentano lo schema di pianta dell'edificio, nell'immagine di attivazione e scelti i "migliori", che vengono poi numerati. I poligoni così ottenuti vengono poi estrusi. La stessa architettura delle DNN viene utilizzata anche per le regole successive, salvo alcune variazioni.

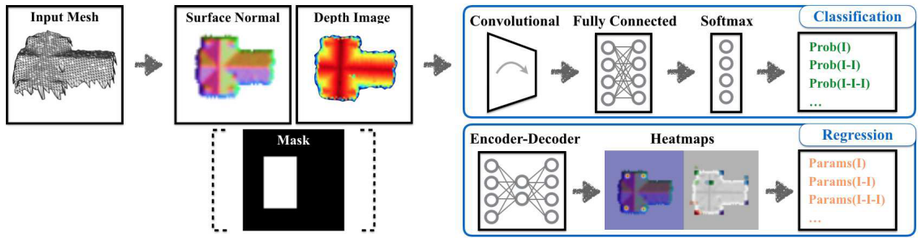


Figura 5.4_6 Classificazione del ramo delle regole e regressione dei parametri geometrici di un neurally-guided procedural model. (Zeng et al. (2018). "Neural procedural reconstruction for residential buildings").

Sebbene il modello non sia sempre impeccabile (ad esempio, in figura 5.5_8, possiamo vedere alcuni errori tipici: classificazione errata dei tipi di fondazione, abbaini o camini mancanti, non riconoscimento di stili architettonici non previsti dalla grammatica), la Neural Procedural Reconstruction ha dimostrato di saper trasformare con successo i dati "rumorosi" ottenuti da sensori in geometrie di qualità CAD.

1 "ResNet": https://andreaprovino.it/resnet. (visitato il 01/11/2020)

2 La funzione softmax, nota anche come softargmax o funzione esponenziale normalizzata, è una generalizzazione della funzione logistica a più dimensioni. Vedi: Goodfellow, I., Bengio, Y., & Courville, A. (2016). 6.2. 2.3 softmax units for multinoulli output distributions. In Deep Learning, MIT Press, 180-184.; Bishop, C. M. (2006). Pattern recognition and machine learning, Springer.

3 Hinton, G. E., & Zemel, R. S. (1994). Autoencoders, minimum description length and Helmholtz free energy. In Advances in neural information processing systems, 3-10.

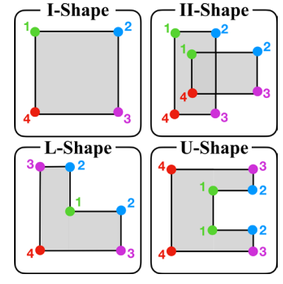


Figura 5.4_5 Numerazione dei vertici dei poligoni che rappresentano lo schema di pianta dell'edificio. (Zeng et al. (2018). "Neural procedural reconstruction for residential buildings").

Figura 5.4_7 Ricostruzioni rappresentative con risultati intermedi. L'immagine satellitare, presa da Google Maps, è mostrata come riferimento sulla sinistra. (Zeng et al. (2018). "Neural procedural reconstruction for residential buildings").

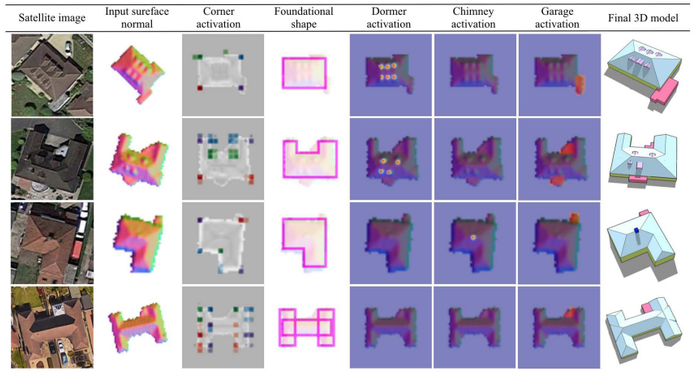


Figura 5.4_8 Esempi di "fallimento" del modello. (Zeng et al. (2018). "Neural procedural reconstruction for residential buildings").

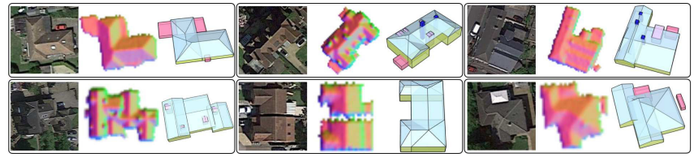
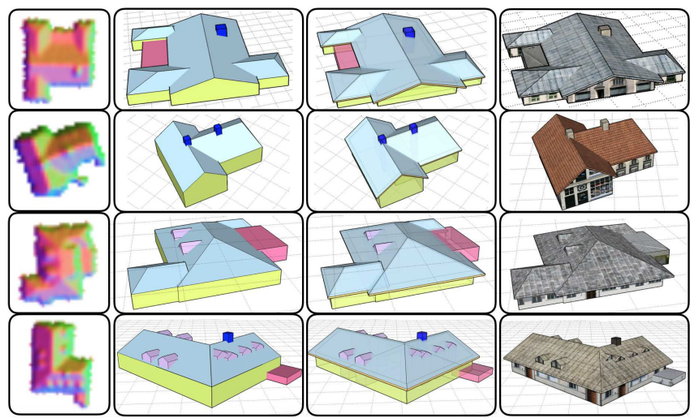


Figura 5.4_9 Fasi evolutive del modello: Da sinistra a destra, un'immagine di input, il modello ricostruito, un modello 3D aumentato e un modello 3D mappato con texture. (Zeng et al. (2018). "Neural procedural reconstruction for residential buildings").



Scheda 8: S.G. + apprendimento per rinforzo

Ruiz-Montiel et al., nell'articolo "Design with shape grammars and reinforcement learning" [1], hanno proposto la combinazione di grammatiche ingenue con l'apprendimento per rinforzo[2], una forma di apprendimento basata sul ricevere una "ricompensa":

"In the absence of knowledge of the utility function, the agent must at least receive some sort of reward or reinforcement that enables it to distinguish between success and failure. Rewards can be received during the agent's activities in the environment or in terminal states that correspond to the end of an episode"[3].

La scelta di questo tipo di apprendimento è dovuta a una serie di considerazioni sui possibili vantaggi rispetto a tecniche di apprendimento supervisionato:

"Unlike supervised techniques, reinforcement learning does not need externally provided examples; it just requires interaction with the environment. Such feature can be very useful when is impractical to obtain representative examples of all the situations in which the agent is involved. Additionally, reinforcement learning allows dealing with uncertainty, making possible to consider more realistic, non-isolated environments"[4].

Dopo aver impostato una grammatica semplice, lo studio prevede una fase di apprendimento in cui viene applicato un algoritmo di apprendimento di rinforzo che guida l'esecuzione della grammatica verso soluzioni realizzabili. I requisiti di progettazione sono formulati come "ricompense" e l'algoritmo impara come ottimizzarli durante la generazione della forma. Va notato, tuttavia, il processo di apprendimento riguarda l'ordine di applicazione delle regole e non le regole stesse. Questa metodologia, che consente la generazione di una grande varietà di progetti realizzabili usando regole relativamente semplici, viene applicata per generare layout schematici di piante di abitazioni unifamiliari conformi a determinate linee

1 Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning".

2 Sutton, R. S. & Barto, A. G. (1998). Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA; Russell, S. (1996). "Machine Learning", in Boden, M. (ed.). Artificial Intelligence, Academic Press.

3 Russell, S. (1996). "Machine Learning", 110.

4 Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning", 234.

guida. L'obiettivo è di dare uno strumento di aiuto nelle prime fasi della progettazione architettonica, offrendo dei punti di partenza fattibili e vari per il processo progettuale. Il processo è articolato in sei fasi:

Fase 1: generazione di un contorno.
Fase 2: etichettatura dello spazio di distribuzione.
Fase 3: posizionamento dei moduli cucina.
Fase 4: posizionamento dei moduli bagno.
Fase 5: etichettatura degli spazi non specializzati.
Fase 6: etichettatura dell'ingresso.

A ciascuna fase sono associate delle regole. Per generare progetti validi, bisogna incorporare una "politica"[1] (policy) in ogni regola, che selezionerà la trasformazione più adatta da applicare in ogni fase al fine di produrre soluzioni fattibili. Nell'articolo è dimostrato come, in assenza di policy, e quindi di sistema di controllo, la grammatica ingenua generi facilmente soluzioni non valide (vedi figure 5.5_2 e 5.5_3). Il processo ha permesso la generazione di un centinaio di progetti, di cui solo 11 hanno violato alcuni dei vincoli posti in partenza.

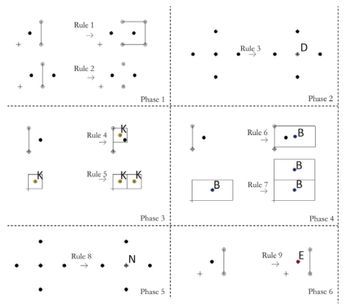


Figura 5.5_1 SG + apprendimento per rinforzo: grammatica utilizzata da Ruiz-Montiel et al.(Fonte: Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning").

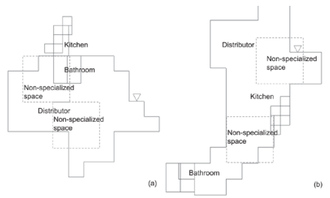


Figura 5.5_2 SG + apprendimento per rinforzo: esempio di schemi generati senza "policy" (Fonte: Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning").

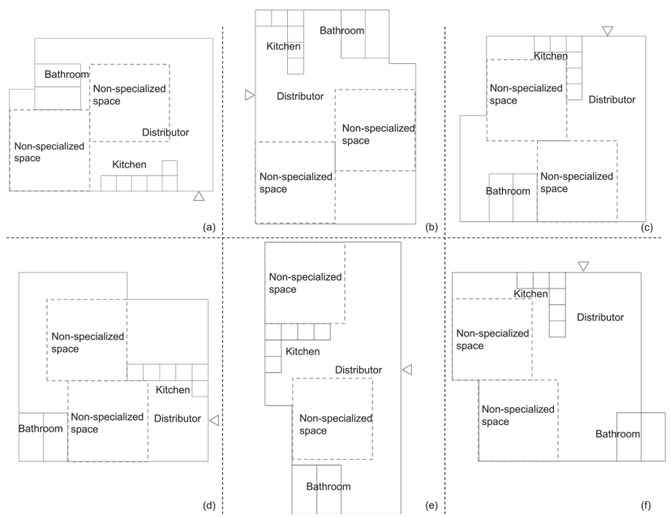
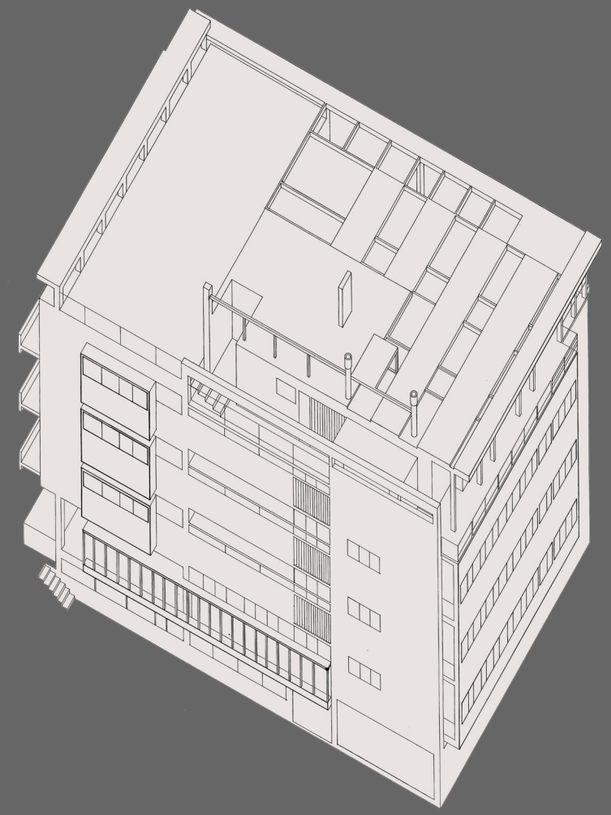


Figura 5.5_3 SG + apprendimento per rinforzo: esempio di schemi generati con "policy" (Fonte: Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning").

1 "The solution to a reinforcement learning problem is a policy, which is basically a function that maps each possible state to an action to be taken when at that state". Ruiz-Montiel et al. (2013). "Design with shape grammars and reinforcement learning", 234.

CASO STUDIO 1

Casa Giuliani Frigerio





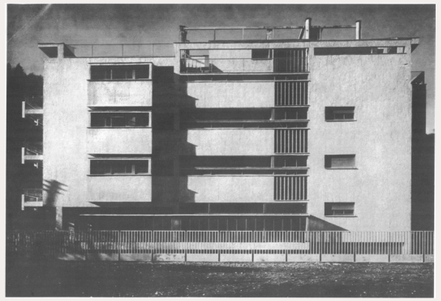
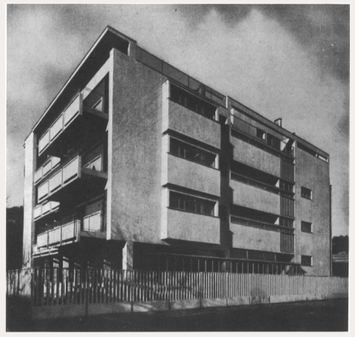



Figura cs1.1_1 Viste di Casa Giuliani Frigerio (Eisenman, P. (1971). "From object to relationship 2: Giuseppe Terragni. Casa Giuliani Frigerio"). Nota: tutte le immagini di questa sezione sono tratte dagli articoli citati ad eccezione delle figure cs1.1_2 e cs1.2_6, ad opera dell'autore. Le immagini originali sono state talvolta modificate.

Casa Giuliani Frigerio

Le grammatiche generative o trasformative inventate da Chomsky nelle sue *Strutture sintattiche*[1] hanno avuto grande impatto sulla cultura architettonica, in particolare attraverso il lavoro di Peter Eisenman.

Nel lavoro di Eisenman due idee nel lavoro di Chomsky sembrano centrali. In primo luogo, la necessità di separare la sintassi dalla semantica; e in secondo luogo, che nella prima è possibile discernere due aspetti: una sintassi di superficie e una sintassi di livello profondo.

Eisenman traspone questi concetti in architettura assumendo la struttura profonda come struttura di riferimento affinché il significato possa essere derivato da una particolare relazione di forme specifiche. Vengono pertanto analizzati metodi di trasformazione per derivare e mettere in relazione forme specifiche con universali formali. Questi dispositivi di trasformazione traducono regolarità formali in forme specifiche.

Casa Giuliani Frigerio rappresenta il banco di prova del modello linguistico proposto da Eisenman e, nel momento in questi si confronta con questo progetto, emergono con chiarezza i limiti e le incongruenze di tale approccio. Nella sua prima analisi formale di Casa Giuliani-Frigerio (1971)[2] infatti Eisenman aveva tentato spiegare il progetto attraverso uno specifico metodo trasformazionale, un metodo cioè che mette in relazione le forme specifiche con una serie di universali formali. Si rende presto conto però del fatto che tale metodo non risulta adeguato all'analisi, arrivando alla conclusione che ciò sia da attribuirsi al ruolo dominante ricoperto dall'intuizione nella progettazione.

Studi successivi, sfociati nel libro *Giuseppe Terragni: Transformations, Decompositions, Critiques*[3], lo portano a dare del progetto una lettura *critico testuale*, che nega i caratteri di stabilità, finitezza e immutabilità dell'architettura, fino a mettere in discussione i concetti stessi di parte e tutto.

Dieci anni dopo la pubblicazione dell'articolo del '71 di Eisenman, U. Flemming conduce un'altra analisi della sintassi di Casa Giuliani Frigerio, avvalendosi dell'algoritmo di Shape Grammar[4].

---

1 Chomsky, N. (1957). *Syntactic Structures*. 's-Gravenhage, Mouton.

2 Eisenman, P. (1971). "From object to relationship 2: Giuseppe Terragni. Casa Giuliani Frigerio", *Perspecta: The Yale Architectural Journal* number 13/14, 36-61.

3 Eisenman, P. (2003). 'Terragni and the Idea of the Critical Text', in *Giuseppe Terragni: Transformations, Decompositions, Critiques* (New York, Monacelli Press.

4 Flemming, U. (1981) "The secret of the Casa Giuliani Frigerio", *Environment and Planning B*,

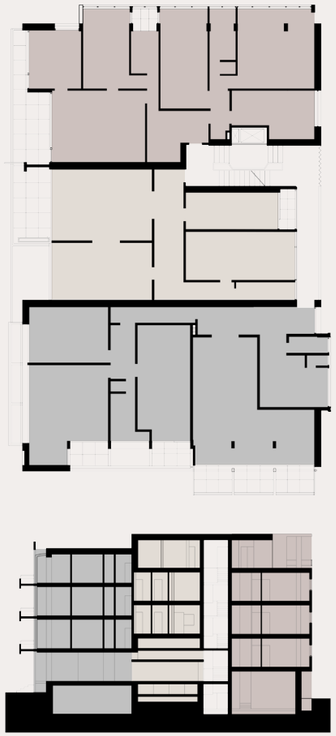


Figura cs1.1_2 Schemi di pianta e sezione di Casa Giuliani Frigerio che evidenziano la tripartizione dell'edificio.

In questo caso studio verranno esaminate entrambe le analisi e si proverà, dalla comparazione tra le due, a mettere in evidenza le differenze sostanziali (sia in termini di metodo che di risultati) e il possibile "vantaggio" che l'analisi condotta con shape grammar ha rispetto all'analisi condotta con mezzi "tradizionali".

cs1.1 Descrizione di Casa Giuliani Frigerio

Casa Giuliani Frigerio (1939-40) è l'ultimo edificio progettato da Terragni prima di partire per il fronte di guerra in Russia, circostanza che lo costrinse a seguire l'andamento dei lavori, poco dopo l'inizio della costruzione, mediante un fitto scambio epistolare con l'amico e collaboratore Luigi Zuccoli.
L'edificio, che occupa un piccolo lotto tra viale Fratelli Rosselli e via Sinigaglia a Como, è caratterizzato da uno schema rettangolare riconducibile a due quadrati sovrapposti, segnalati dai setti che attraversano trasversalmente il volume. Elevato su quattro piani fuori terra, ha un piano rialzato con due alloggi molto arretrati sul perimetro a nord ovest per lasciare spazio ad un ampio porticato di accesso. Gli appartamenti, tre per ogni piano superiore, sono posti su livelli differenziati: questo produce una disarticolazione dello schema del parallelepipedo di base leggibile anche in facciata, attraverso il gioco di allineamenti e disallineamenti delle aperture. La disposizione sfalsata dei piani consente al corridoio che dà accesso all'appartamento sud, di incrociarsi con l'appartamento centrale, assicurando aria e ventilazione dagli sbalzi ricavati. Due alloggi sono situati sulla stessa quota, mentre un terzo, servito da un ballatoio esterno, è posto ad una quota successiva.
"Sin dai primi disegni, Terragni, dispose tre appartamenti per piano, creando però uno sfalsamento di mezzo piano fra i due alloggi verso nord e il terzo posto verso sud: l'accesso a quest'ultimo era previsto dal pianerottolo intermedio per mezzo di un ballatoio. Nella parte centrale della facciata ovest, tale ballatoio appariva come una fascia sormontata da una finestra a nastro e sovrastante un'apertura in lunghezza, per consentire il riscontro d'aria all'alloggio. L'appartamento dell'ultimo piano era una vera e propria villa, circondata da terrazzi a diversi livelli"[1].

volume 8, 87-96.

1 Ciucci, G. (1996). *Giuseppe Terragni : Opera completa*, Electa, 585.

cs1.2 Eisenman: "Da oggetto a relazione"

Nel suo articolo "From object to relationship 2: Giuseppe Terragni. Casa Giuliani Frigerio"[1] Petetr Eisenman utilizza l'analisi formale come dispositivo per riconoscere una struttura sintattica a doppio livello nell'opera di Terragni.
Viene indagato in particolare il metodo di trasformazione specifico utilizzato per spiegare la relazione tra struttura profonda e aspetti superficiali dell'architettura attraverso l'analisi di una forma specifica, rappresentata da Casa Giuliani Frigerio stessa, utilizzando il confronto con la Casa del Fascio come metodo di indagine. Tale metodo di trasformazione, che Eisenman riconosce nell'*ambiguità pittorica* (o concettuale), consiste nella contrapposizione tra oggetto/realtà percepita e relazioni formali astratte sottese. Nel caso della Giuliani-Frigerio tale ambiguità risiede nel fatto che è possibile interpretarla sia come spazio planare stratificato, sia come spazio volumetrico. Secondo Eisenman infatti, nel lavoro di Terragni, vi è una sostanziale sovrapposizione di due concezioni basiche e opposte dello spazio: la prima considera lo spazio come sottrattivo o ricavato da un solido, è cioè considerato metaforicamente scavato da un volume solido astratto; la seconda, che ha antecedenti del Rinascimento, considera lo spazio come additivo, costituito da una serie di strati impliciti. Lo spazio sottrattivo implica un centro ed è di concezione centripeta; lo spazio additivo si occupa maggiormente della periferia, con bordi e angoli, ed è di concezione centrifuga. Quindi a livello concettuale, lo spazio nel suo stato più neutro è visto come pieno-positivo o vuoto-negativo.
Nell'interpretazione di Eisenman, l'edificio viene considerato sostanzialmente "esternalizzato", vale a dire che la struttura spaziale esterna viene interpretata come in relazione principalmente con l'ambiente esterno piuttosto che con la struttura interna. Sostiene infatti non solo che nella Giuliani-Frigerio vi è poco spazio interno, ma anche che la struttura funzionale interna sembra derivare poco della sua organizzazione dalle facciate esterne. Allo stesso modo, poco della struttura interna si manifesta sugli aggetti e le distorsioni delle facciate. Per questo motivo l'analisi si concentra sullo studio delle facciate, viste come sistemi indipendenti, per quanto tutte relazionate al "volume interno".
Un'altra questione importante è la concezione del rapporto tra l'osservatore e l'edificio come determinato dall'orientamento concettuale dell'edificio stesso. Eisenman osserva che, a differenza della Casa del Fascio, che ha un orientamento principalmente frontale, nella Giuliani-Frigerio coesistono sia l'orientamento obliquo

1 Eisenman, P.D. (1971). "From object to relationship 2: Giuseppe Terragni. Casa Giuliani Frigerio".

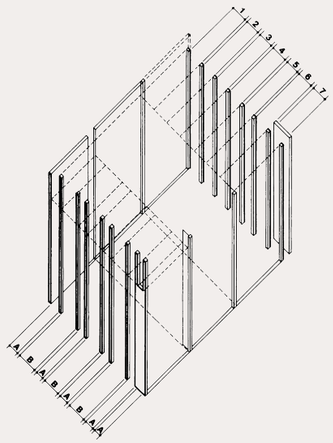


Figura cs1.2_1 Struttura delle campate

che frontale e, a causa dell'importanza dell'articolazione degli angoli nell'intenzione formale, l'orientamento "obliquo" sembra prevalente.
Ciò porta Eisenman a una seconda distinzione tra i due sistemi formali. Nella Casa del Fascio l'enfasi frontale - la stratificazione dello spazio da un dato frontale - è considerata principalmente in relazione al contesto specifico; nel rapporto tra l'edificio e la piazza adiacente e la cattedrale. Nella Giuliani-Frigerio invece l'enfasi obliqua è meno chiara e meno risolta in termini di contesto immediato. Ad esempio, se le facciate nord e est vengono lette come "solide" e le facciate sud e ovest vengono lette come "vuote", viene stabilita una diagonale risultante che non sembra rispondere alle condizioni esistenti del sito quanto piuttosto derivare dalla particolare sintassi di un'ambiguità tra planare e volumetrico.

Struttura dell'analisi di Eisenman

Eisenman propone una lettura dell'opera attraverso l'analisi non solo dell'edificio in sé ma di tutto il processo progettuale. Dall'evoluzione degli schemi progettuali deduce il rapporto tra fatto fisico e struttura concettuale latente. Tale rapporto viene visto da Eisenman come esplicitato dall'insieme di trasformazioni che relazionano il solido concettuale interno al sistema esterno delle facciate. Questo lo porta a strutturare l'analisi attraverso la lettura delle singole facciate viste come sistemi indipendenti, ciò anche sulla base del particolare trattamento che Terragni fa delle soluzioni d'angolo per cui i piani che definiscono l'involucro esterno dell'edificio non si toccano mai.

Facciata Nord

La struttura formale della facciata nord è utile a chiarire l'ipotesi di ambiguità concettuale.
Innanzitutto, la distribuzione dei pilastri lungo la facciata segue uno schema AbAb di campate alternate, a differenza della facciata sud dove le campate sono uguali (Fig. cs1.2_1). Questo sembra dichiarare non solo una discontinuità nella direzione longitudinale dell'edificio, ma anche negare l'unitarietà dell'oggetto architettonico.
In secondo luogo, il volume aggettante sulla facciata potrebbe essere letto come l'estrusione del volume interno oltre la linea dei pilastri. Tuttavia, questa estensione volumetrica non interessa l'intera facciata e, nell'angolo nord-est, non è "chiusa". Questa condizione consente ad Eisenman una doppia lettura: o la facciata è stata estesa, in modo additivo, come una sequenza di piani, oppure il bordo esterno è stato tagliato per rivelare un volume "solido" interno.

Eisenman descrive pertanto la facciata nord come un piano piegato e vede l'uso di questa soluzione da parte di Terragni come in relazione con la struttura concettuale. Il piano piegato funziona in due modi per articolare l'intenzione di contrastare una lettura di solido scavato con quella di una sequenza di strati spaziali. In primo luogo, se la facciata nord viene considerata come un solido concettuale che è stato scavato, il bordo superiore orizzontale della superficie piegata deve essere visto come una cornice che segna i limiti del solido iniziale che, con la superficie del blocco dei tre piani tipici, forma un piano verticale concettuale da cui tutte le rientranze possono essere lette come scavi (Fig. cs1.2_3).
Una seconda lettura, quella di una stratificazione di piani, è determinata dalla particolare articolazione di questa superficie ed assume la stessa valenza. La condizione di taglio è rafforzata dalle finestre a fessura stretta che appaiono solo in una campata e insieme alle fessure orizzontali sul volume aggettante danno l'impressione che questo si spostato in direzione nord-est (Fig. cs1.2_4).

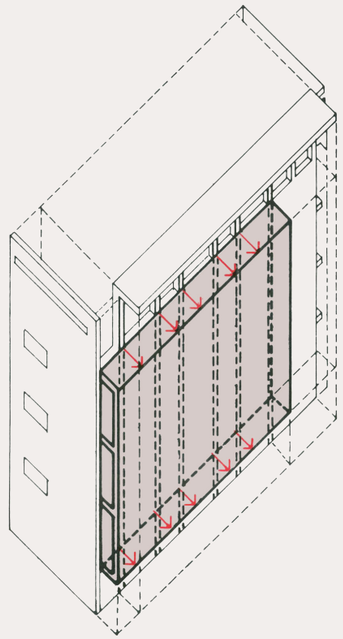


Figura cs1.2_2 Estrusione del volume aggettante.

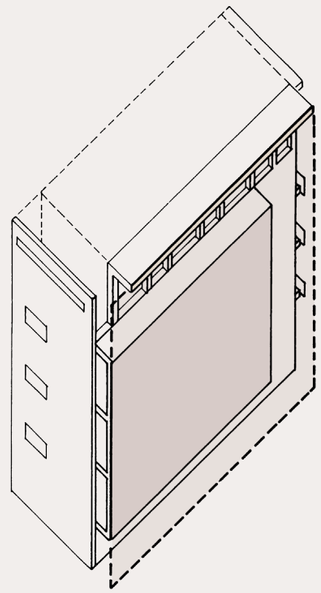


Figura cs1.2_3 Scavo del volume aggettante dal solido concettuale.

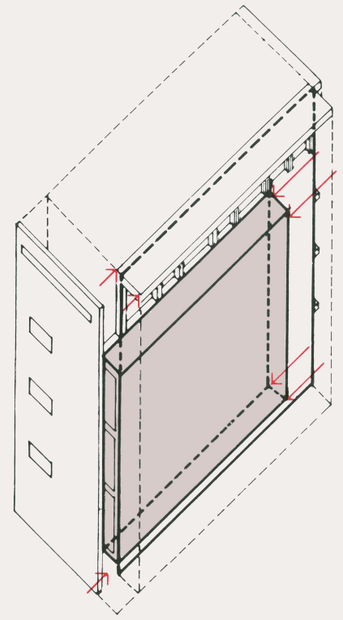


Figura cs1.2_4 Slittamento del volume aggettante.

Facciata Est

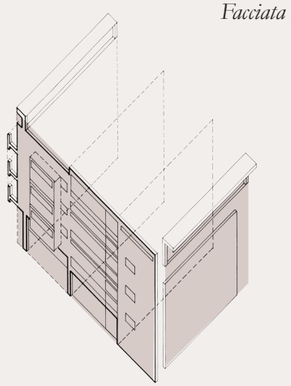


Figura cs1.2_5 Rapporto tra i piani delle facciate est e nord.

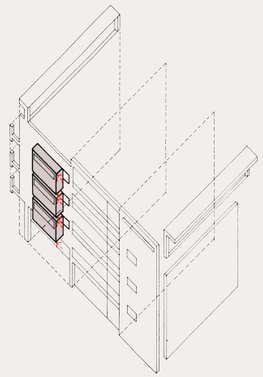


Figura cs1.2_7 Slittamento dei volumi sulla facciata.

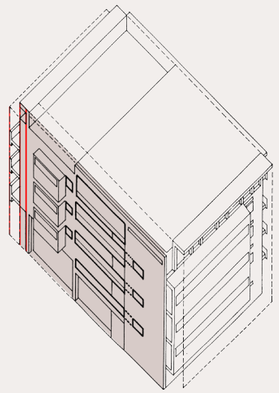


Figura cs1.2_8 Rapporto tra i piani delle facciate est e sud.

La facciata est viene interpretata sia come un piano di riferimento su cui è marcata la stratificazione laterale del volume da sud a nord (Fig. cs1.2_5), sia come riferimento per il solido concettuale originale.

Eisenman osserva inoltre che mentre nella Casa del Fascio, dove la dialettica è rivelata all'interno del piano della facciata stessa attraverso l'apposizione dei piani all'interno della struttura reticolare, nel caso di Casa Giuliani Frigerio è piuttosto agli angoli, articolati come un incontro di piani piuttosto che come un bordo di un solido volumetrico, che la seconda lettura diventa evidente.

Dunque, mentre nella Casa del Fascio la struttura concettuale è articolata attraverso la dialettica tra solido eroso e griglia reticolare, in casa Giuliani-Frigerio la dialettica è tra solido eroso e una sequenza di piani, con una corrispondente soppressione delle lettura reticolare.

Così mentre la Casa del Fascio sopprime la struttura diagonale per le relazioni frontali, in Giuliani-Frigerio ricerca il punto di vista "obliquo".

Sulla facciata est le aperture principali sono trattate come bande continue attraverso la facciata, dando una continuità implicita al piano. Anche quando ci sono aperture isolate, come nella campata di destra, continuano la linea implicita della fascia. In questo vano le finestre sono posizionate leggermente lontano dalla linea dei pilastri. Questa posizione le libera da una possibile interpretazione come a segnare questa linea sulla superficie, impedendo così una lettura reticolare della facciata.

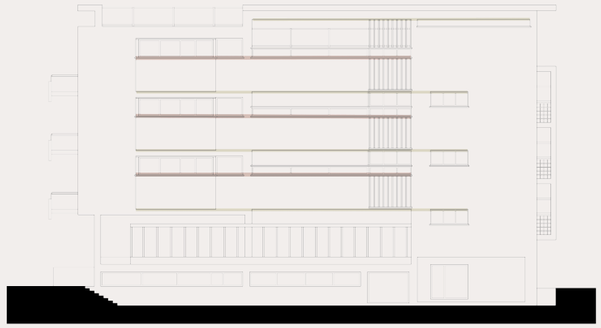


Figura cs1.2_6 Allineamenti sulla facciata est.

Nella campata di sinistra è presente un aggetto volumetrico che non è un volume unico, ma è ripartito in tre scatole separate. Questi solidi sembrano slittare sulla facciata a causa delle le aperture orizzontali che, come sulla facciata nord, sembrano la traccia del loro movimento. Ciò conferisce a questa una qualità planare (Fig. cs1.2_7).

Da sud-est, la sezione della parete verticale si estende come un piano oltre la linea del pilastro che Eisenman legge come l'elemento che segna l'angolo del volume interno (Fig. cs1.2_8). I balconi del lato sud slittano dietro questo piano in modo tale da metterlo ulteriormente in evidenza. Da sud la natura planare della facciata est è ulteriormente enfatizzata dall'angolo in alto a destra dove il piano è articolato da un leggero gap che rompe di nuovo l'angolo "solido".

Eisenman, nella sua analisi, definisce le facciate sud e ovest come le due facciate "aperte", vale a dire quelle in cui, a differenza delle facciate est e nord, è possibile leggere la struttura reticolare ed è più esplicita la relazione interno-esterno. Della facciata sud (Fig. cs1.2_9) viene rilevata la somiglianza con la facciata d'ingresso della Casa del Fascio (Fig. cs1.2_10) per via della simile struttura tripartita con la parte "piena" della facciata invertita di posizione (nella Casa del Fascio è sulla destra, nel Giuliani-Frigerio è sulla sinistra). Questo elemento pieno, rispetto alla Casa del Fascio, è ridotto alla larghezza di una sola campata e va a formare con il bordo del piano orizzontale del tetto una cornice.

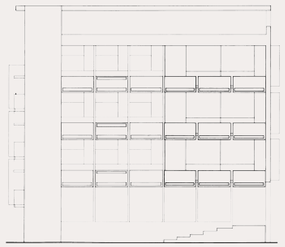


Figura cs1.2_9 Facciata sud.

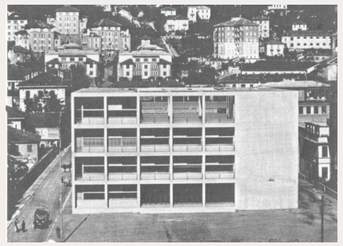


Figura cs1.2_10 Casa del Fascio.

Inoltre, mentre nella Casa del Fascio gli elementi verticali e orizzontali sono collocati nel piano della facciata, nella Giuliani-Frigerio viene utilizzata una struttura formale più ambigua, dove non solo si riscontra la solita ambiguità concettuale solido/planare, ma anche i piani di riferimento cambiano a seconda dei punti di vista, dando luogo a diverse letture.

Secondo Eisenman, infatti, se una possibile interpretazione vede questa cornice come il bordo di un solido scavato, è possibile anche intenderla come un piano di riferimento rispetto al quale gli elementi si proiettano in avanti o rientrano per contrassegnare una serie di piani verticali (Fig. cs1.2_11). Rispetto a questo piano di riferimento dunque le colonne sono arretrate per contrassegnare un layer e i balconi si proiettano in avanti per segnarne un altro.

Tuttavia, quando si passa da un punto di vista frontale a obliquo, la lettura cambia:

Facciata Sud

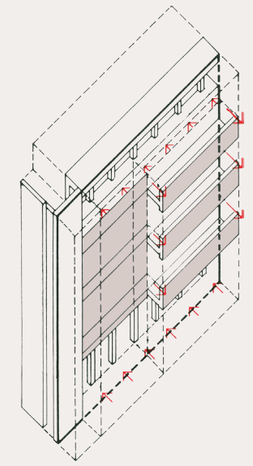


Figura cs1.2_11 Stratificazioni di piani sulla facciata sud.

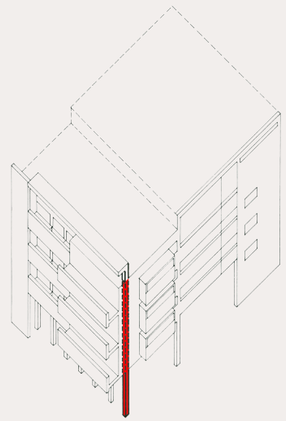


Figura cs1.2_12 Vista da sud-est.

se visto da sud-est, non è il piano della parete verticale di sinistra e del tetto che si legge come riferimento, ma piuttosto è il piano delle colonne, che dal punto di vista obliquo, diventa il riferimento principale essendo è l'unica linea verticale continua (Fig. cs1.2_12).

Facciata Ovest

Per l'analisi della facciata ovest Eisenman si concentra sulla lettura delle soluzioni d'angolo, dove appare più evidente l'intenzione di trattarla come una facciata "aperta". In questo caso la dialettica dei volumi solidi e della stratificazione planare viene considerata meno importante dell'ambiguità sviluppata dalla marcatura della stratificazione verticale (Fig. cs1.2_13). A nord-ovest la linea dei balconi e il bordo del tetto sporgono oltre l'angolo della facciata nord, impedendo così di nuovo qualsiasi lettura volumetrica (Fig. cs1.2_14).

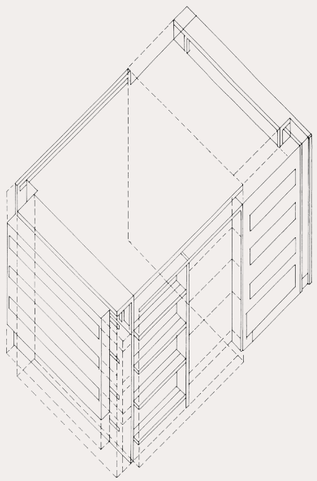


Figura cs1.2_13 Stratificazione verticale sulla facciata ovest.

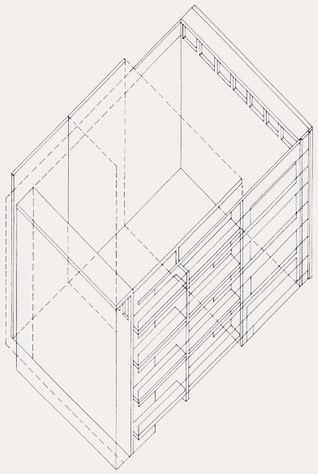


Figura cs1.2_14 Rapporto tra le facciate nord e ovest.

cs1.3 Eisenman: scomposizioni e letture critico testuali

Nell'articolo del '71 Eisenman aveva tentato dunque di mostrare uno specifico metodo trasformazionale, avvertendo però l'inadeguatezza di tale metodo per l'analisi dell'edificio, arriva alla conclusione che ciò sia da attribuirsi al ruolo dominante ricoperto dall'intuizione nella progettazione.

Nel 2003 pubblica il libro *Giuseppe Terragni - tranformation, decomposition and critiques*[1], dove torna ad affrontare l'argomento, riproponendo un confronto tra Casa del Fascio e Casa Giuliani Frigerio nel tentativo di delineare un unico approccio metodologico per la lettura di entrambe le opere. Tuttavia, constatata l'impossibilità di applicare a Casa Giuliani Frigerio la matrice generale di carattere critico testuale (e in particolare le *trasformazioni*) costituita dalle categorie sviluppate nella lettura di Casa del Fascio, introduce una nuova categoria: la *scomposizione*.

"Mentre per definizione la trasformazione dipende dalla leggibilità delle convenzioni che vengono modificate, la scomposizione esige lo sviluppo di una nuova cornice analitica, poiché l'edificio rivela la disgregazione delle condizioni convenzionali precedenti solo dopo che è stata intrapresa l'analisi"[2].

Laddove nella trasformazione si assume che vi sia un *grado zero* dell'architettura (tipologia, programma, linguaggio formale, luogo etc.) a partire dal quale le strategie compositivo-trasformazionali possono avere luogo, nella scomposizione non esiste un grado zero. Gli elementi che compongono Casa Giuliani Frigerio sono tracce di uno sviluppo discontinuo privo di un'origine interna dalla quale ricostruire una narrativa. La scomposizione è un processo indeterminato che dà luogo a relazioni instabili.

La metodologia di analisi adottata tiene conto sia del fatto che il significato non può sempre essere razionalizzato come effetto di una causalità e di uno sviluppo lineare, sia del fatto che i due edifici di Terragni non possono essere spiegati unicamente da tipologie formali estetiche e funzionali o da analisi concettuali. La struttura analitica dello studio, determinata dalla natura specifica dei due edifici, non è basata su rapporti funzionali o estetici, ma su rapporti *testuali critici*[1].

La lettura critico testuale cerca di mettere in discussione l'idea che l'architettura presenti sempre attributi finiti, letture immutabili e un'origine stabile. Il *testo critico* è infatti "(…)qualcosa che pertiene specificamente all'oggetto architettonico, che mina la nozione di attributi finiti e può essere articolato o compreso solo attraverso un diverso linguaggio interpretativo. Un simile linguaggio interpretativo si concentra su notazioni quali sovrapposizione residuale, traccia, alternanza, oscillazione e slittamento"[2].

Casa del Fascio e Casa Giuliani Frigerio sono esempi di testi critici architettonici poiché le letture convenzionali delle loro facciate, piante e sezioni non sono stabili. Rappresentano un allontanamento dall'architettura basata su gerarchia, unità, sequenza, progressione e continuità.

Pertanto, il testo si divide in due parti, la prima dedicata all'analisi di Casa del Fascio e la seconda a quella di Casa Giuliani Frigerio. Anche in questo caso quest'ultima viene studiata attraverso l'analisi separata delle quattro facciate in quanto lo spazio interno è ritenuto meno importante delle facciate come generatore di materiale testuale. A causa della scarsa relazione formale tra interno ed esterno infatti piante e sezioni risultano indipendenti dalle facciate. Secondo Eisenman in Casa Giuliani Frigerio manca un rapporto tra interno ed esterno e, a differenza di Casa del Fascio, tra le facciate vi è una relazione *oscillante*, non narrativa, anziché un rapporto *alternante* narrativo. *Oscillante* e *alternante* sono condizioni testuali legate alla stratificazione interna dell'edificio.

Tra le proprietà volumetriche di un edificio Eisenman infatti individua innanzitutto la direzionalità: un edificio ha una direzione longitudinale (la direzione dominante, la stratificazione verticale è parallela al piano assunto come riferimento) e una trasversale (la direzione secondaria, la stratificazione verticale è ortogonale al piano di riferimento). In caso di direzione incrociata nessuna direzione è dominante, tende

1 Peter Eisenman (2003), *Giuseppe Terragni: Transformations, Decompositions, Critiques,* Monacelli Press; M. Baiocchi, A. Coppola (Trad.)(2004) *Giuseppe Terragni: trasformazioni, scomposizioni, critiche*, Quodlibet.
2 Ibid., 155.

1 Ibid., 10-11
2 Ibid., 11

per questo a favorire una lettura obliqua o angolare. Sebbene Casa del Fascio abbia un volume quadrato, non siamo in presenza di una direzione incrociata. Vi è una direzione *alternante* in quanto vi è una distinzione tra direzione longitudinale e trasversale: la facciata di entrata infatti segnala la direzione longitudinale, mentre la griglia su di essa rivela quella trasversale. In Casa Giuliani Frigerio vi è invece, secondo Eisenman, una direzione *oscillante*, cioè vi è un costante *avanti-indietro* mai risolto, mentre in una direzione alternante vi è un *avanti-indietro* con intervalli di stabilità. Pertanto, in Casa Giuliani Frigerio manca una stratificazione concettuale interna.
Le condizioni testuali *oscillante* e *alternante* sono quindi definite dalla *lettura primaria* di un edificio, che definisce il rapporto tra angolo e facciata. Questa lettura, fondamentalmente gerarchica nell'architettura classica e dialettica in quella moderna, nei due progetti di Terragni perde stabilità. In Casa Giuliani Frigerio, infatti, l'angolo dà luogo a letture oscillanti, irrisolte, mentre in Casa del Fascio il rapporto obliquo-frontale è risolto in ogni facciata da una serie di letture alternanti. Qui troviamo un'alternanza di punti di vista privilegiati, mentre in Casa Giuliani Frigerio non vi è mai un punto di vista privilegiato e le viste oblique indicano frammentazione e separazione, come testimoniato dalle fratture in corrispondenza degli angoli. Le letture oblique e frontali non sono concettualmente legate le une alle altre, le facciate sembrano infatti appartenere a edifici diversi.
Infatti, mentre in Casa del Fascio vi sono sia relazioni interdipendenti tra angoli e facciate che rapporti indipendenti (cioè viste frontalmente le facciate sono indipendenti ma la lettura obliqua rivela l'interdipendenza attraverso gli angoli), in Casa Giuliani Frigerio vi è un minore rapporto tra letture indipendenti e interdipendenti. Le notazioni angolari date dalla lettura obliqua infatti non possono essere lette nelle viste frontali. Vi sono elementi, come ad esempio la cornice piegata sulla facciata nord, che possono essere compresi concettualmente solo da una lettura obliqua. Non vi è tuttavia una comprensione narrativa, che stabilisce il rapporto di una parte col tutto, né è inserita in una lettura alternante, in cui ogni volta che una lettura si suggerisce come parte di un tutto, quel tutto viene contraddetto da un'altra lettura. In Casa Giuliani Frigerio i concetti stessi di parte e tutto sono messi in discussione.

cs1.4 Flemming: "Il segreto di Casa Giuliani Frigerio"

Nel suo articolo "The secret of the Casa Giuliani Frigerio"[1] U. Flemming prova a descrivere le caratteristiche formali di Casa Giuliani Frigerio di Terragni attraverso una grammatica di forma parametrica.
Il lavoro si concentra sull'analisi dei tre piani intermedi dell'edificio e introduce una grammatica che produce una pianta parziale di questi piani sotto forma di una sezione orizzontale attraverso le facciate. Le forme sono realizzate in un sistema cartesiano di coordinate e i simboli x e y sono usati per indicare le coordinate di un punto nel sistema. Invece di regole di forma, una grammatica di forma parametrica contiene schemi di regole di forma; da qualsiasi schema di questo tipo è possibile ricavare una regola di forma specifica tramite un'assegnazione di valori, *g*, ai parametri nello schema. Tutti gli schemi usati nella grammatica sono completamente parametrizzati; cioè, entrambe le coordinate x e y dei punti finali di ciascuna linea massimale in uno schema sono date come parametri. Qualsiasi assegnazione *g* deve mantenere gli allineamenti dei punti paralleli all'asse x o all'asse y come vengono visualizzati dalle specifiche grafiche degli schemi.
Con una grammatica di forma parametrica, un corpus contenente esattamente una forma, σ, può sempre essere prodotto da una grammatica di forma costituita da una forma iniziale arbitraria I^* e da una singola regola $I^* \rightarrow \sigma$. Per diventare interessante, tuttavia, una grammatica generativa deve dimostrare come σ può essere costruita a partire da regole di base. La grammatica costruita da Flemming esprime tali principi attraverso schemi regole che sono semplici in doppio senso: (1) ogni schema definisce solo un intervallo molto limitato di proprietà e (2) influenza solo una piccola porzione di una forma.
Grammatiche di questo tipo producono, attraverso la natura combinatoria del processo di generazione, forme diverse da σ. Questo catalogo di soluzioni si sottrae alle variazioni causate da rotazioni, riflessioni, differenze dimensionali o numeri variabili di colonne. Ogni forma nella figura dovrebbe quindi essere vista come rappresentativa di un'intera classe di forme che differiscono l'una dall'altra solo per le proprietà trascurate nel catalogo. La grammatica produce sempre contrasti locali agli angoli o all'interno di una facciata, ma non impedisce la ripetizione delle configurazioni di pareti diverse agli angoli o la creazione di simmetrie rotazionali. Di conseguenza, non tutte le forme generate mostrano la gamma di contrasti riscontrati nel design di Terragni.

1 U. Flemming, U. (1981). "The secret of the Casa Giuliani Frigerio", *Environment and Planning B*, volume 8, 87-96

Struttura dell'analisi di Flemming

L'analisi proposta da Flemming è stata modellata sulla grammatica palladiana descritta da Stiny e Mitchell[1], che dividono il disegno in pianta in fasi distinte e utilizzano punti etichettati per segnalare la transizione da uno stadio all'altro. Nello specifico Flemming distingue il processo in sei fasi:
Fase 1: definizione delle colonne
Fase 2: definizione delle pareti
Fase 3: etichettatura delle pareti
Fase 4: sviluppo delle pareti
Fase 5: collegamento delle pareti
Fase 6: design delle finestre

Fase 1: definizione delle colonne

La forma iniziale *I* (figura cs1.4_1(a)) è data dalla sezione orizzontale di una colonna rettangolare. Il suo angolo inferiore sinistro coincide con l'origine del sistema di coordinate.
Lo schema 1 di figura cs1.4_1(b) genera da *I* quattro file di colonne disposte lungo i bordi di un rettangolo o, in una realizzazione tridimensionale, le facce di un cubo nello spazio. Lo schema 2 inserisce una colonna aggiuntiva in una forma generata dallo schema 1. Entrambi gli schemi introducono forme ausiliarie sotto forma di frecce i cui punti centrali sono etichettati come A o B. Un'applicazione dello schema 1 a *I* che è seguita da quattordici applicazioni dello schema 2 produce la forma etichettata Sx nella figura cs1.4_1(c). Questa forma rappresenta non solo le colonne nella facciata nord e sud, ma anche le estremità orientali e occidentali delle pareti portanti di Casa Giuliani Frigerio.

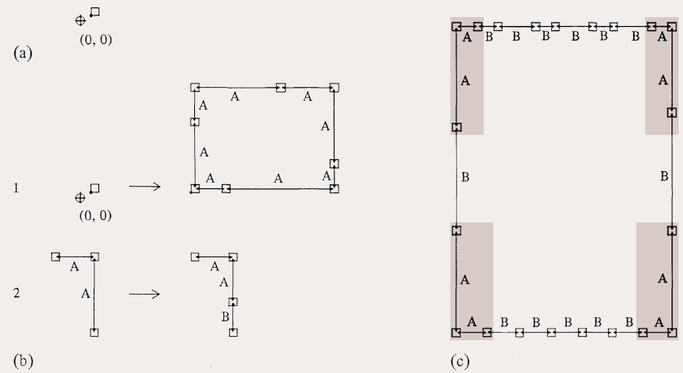


Figura cs1.4_1: (a) La forma iniziale *I*, (b) schemi di definizione delle colonne e (c) forma Sx generato da *I* per gli schemi di definizione delle colonne.

1 Stiny, G., & Mitchell, W. J. (1978). "The palladian grammar", Environment and planning B: Planning and design, 5(1), 5-18.

Come evidenziato in figura cs1.4_1(c), l'applicazione delle regole 1 e 2 porta ad una etichettatura delle campate tale per cui le campate d'angolo (etichettate con *A*) sono distinte dalle altre (*B*). Queste ultime non sono interessate da trasformazioni successive.

Fase 2: definizione delle pareti

Gli schemi mostrati in figura cs1.4_2(a) aggiungono muri agli angoli della forma generata durante la fase precedente. Ad ogni colonna d'angolo in tale forma, due campate si incontrano ad angolo retto. Gli schemi 3 e 4 creano una parete riempiendo una di tali campate. Lo schema 4 posiziona un muro aggiuntivo davanti alla seconda campata. Due applicazioni di ogni schema di definizione della parete trasformano la forma S1 nella forma S2 mostrata nella figura cs1.4_(b).
Gli otto schemi utilizzati nelle fasi 2-4 generano esattamente quattro diverse configurazioni dei muri nell'angolo della forma. Questi schemi potrebbero essere sostituiti da altri che produrrebbero i risultati delle fasi 2-4 in un unico passaggio. Flemming tuttavia preferisce usare schemi più semplici per isolare i principi trovati per l'analisi e dimostrare come l'apparente complessità dell'edificio può essere spiegata attraverso gli effetti combinati di operazioni molto semplici.

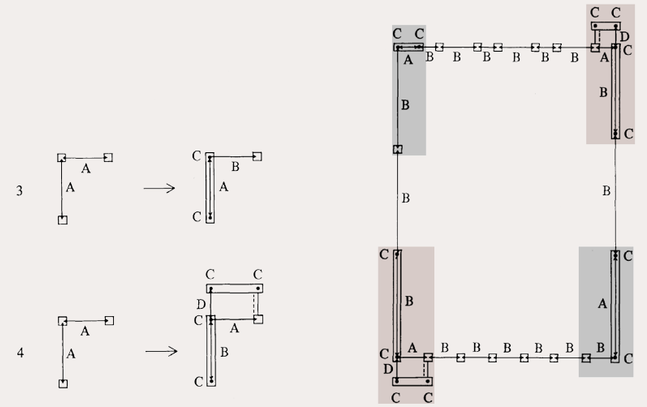
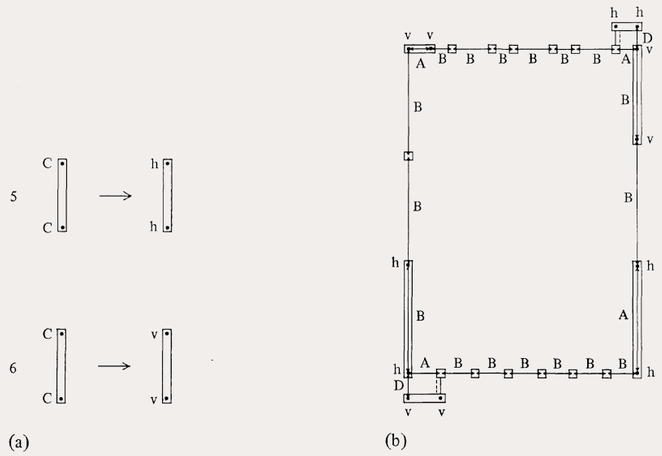


Figura cs1.4_2 (a) Schemi di definizione della parete e (b) forma S2 generata dalla forma S1 dagli schemi di definizione della parete.

Fase 3: etichettatura delle pareti

Gli schemi di figura cs1.4_ 3(a) attribuiscono le etichette *h* o *v* ai muri. I muri con etichette h sono chiamati orizzontali e i muri con etichette v sono chiamati verticali. Sono necessarie tre applicazioni per ciascuno schema di etichettatura per trasformare la forma S2 nella forma S3 mostrata nella figura cs1.4_ 3(b). I punti etichettati con C vengono ri-etichettati con h e v a seconda dello sviluppo in alzato. Anche in questo caso le regole producono un pattern alternato che Flemming vede come caratterizzante la struttura sintattica della composizione.
La Figura cs1.4_ 4(a) mostra gli schemi che modificano alcune pareti etichettate.

Figura **cs1.4_ 3** (a) Schemi di etichettatura del muro e (b) forma S3 generata dalla forma S2 dagli schemi di etichettatura del muro.



Fase 4: sviluppo delle pareti

Lo schema 7 estende una parete orizzontale creata dallo schema 3 oltre la colonna d'angolo. Una parete orizzontale posizionata dallo schema 4 di fronte a una campata viene estesa dallo schema 9 su almeno una campata adiacente. Gli schemi 8 e 10 impediscono la modifica delle pareti verticali (notare come le etichette A e B sono utilizzate come dispositivi di controllo). La forma S3 viene trasformata nella forma S4 nella figura 4 cs1.4_ 4(b) esattamente con un'applicazione di ogni schema di sviluppo del muro. Gli schemi 7-10 modificano puntualmente le quattro pareti le cui campate risultano ancora etichettate con A dagli schemi precedenti

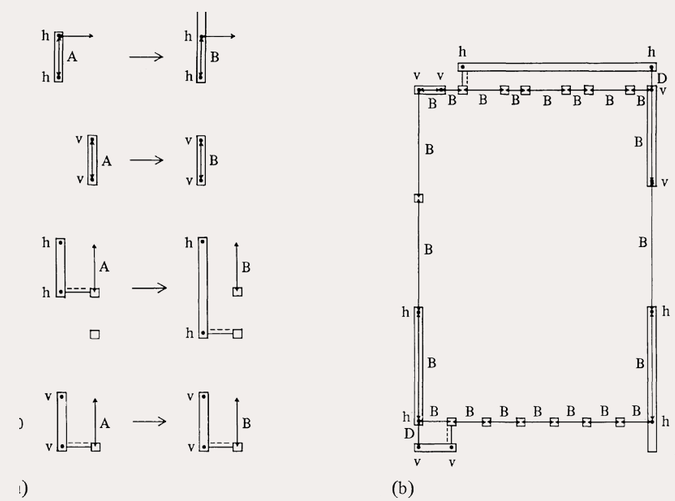


Figura cs1.4_ 4 (a) Schemi di sviluppo del muro e (b) forma S4 generata dalla forma S3 dagli schemi di sviluppo del muro.

Fase 5: collegamento delle pareti

Gli schemi in figura cs1.4_5 collegano i muri che sono stati generati nelle fasi precedenti attraverso linee continue che definiscono il confine esterno di balconi o lastre del pavimento che non terminano in una parete solida e linee tratteggiate che indicano la posizione di pannelli trasparenti o opachi che vanno dal pavimento al soffitto e chiudono gli spazi lasciati tra le pareti solide.

Le condizioni fornite per alcuni parametri nei primi tre schemi della figura cs1.4_5 indicano che la formulazione di questi schemi è abbastanza generale. Ad esempio, le pareti verticale e orizzontale nello schema 11 possono essere posizionate tra le colonne o davanti alle colonne e le posizioni delle due pareti sono indipendenti l'una dall'altra. Le stesse possibilità esistono per la parete orizzontale nello schema 12 e la parete verticale nello schema 13.

Si noti tuttavia che uno spazio chiuso dallo schema 12 o dallo schema 13 deve estendersi su più di una campata, mentre lo schema 11 chiude solo spazi che sono grandi esattamente una campata.

Si noti inoltre che gli schemi nella figura 5 possono essere utilizzati solo per collegare pareti con etichette distinte. Per trasformare la forma S4 nella forma S5 nella figura cs1.4_6, gli schemi 11 e 14 devono essere usati due volte e gli schemi 12 e 13 una volta. Gli schemi 11-14 definiscono puntualmente le soluzioni di chiusura degli spazi tra le pareti.

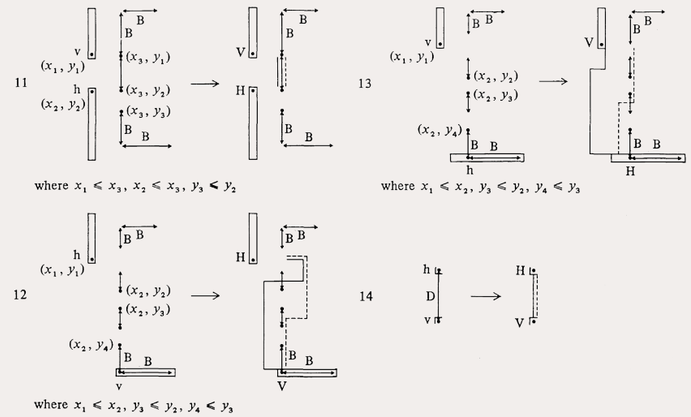


Figura **cs1.4**_5 Schemi di connessione dei muri.

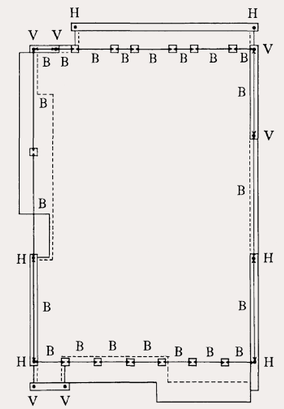


Figura **cs1.4**_6 Forma S5 generata dalla forma S4 dagli schemi di connessione delle pareti.

Fase 6: design delle finestre

La figura cs1.4_7(a) mostra schemi che posizionano le finestre nelle pareti e ne rimuovono le etichette.

Le pareti orizzontali devono ricevere finestre attraverso schemi 15-17; si deve comprendere che qualsiasi schema di questo tipo può essere applicato solo a un muro con una certa dimensione minima e che la lunghezza della finestra deve superare la metà della lunghezza del muro.

Un muro verticale può rimanere senza finestre (schema 18). Se, tuttavia, una finestra viene posizionata in tale muro (schema 19), la lunghezza della finestra deve essere inferiore alla metà della lunghezza del muro. Lo schema 20 rimuove le frecce e le etichette rimanenti.

La forma S* nella figura cs1.4_7(b) è generata dalla forma S5 attraverso gli schemi di progettazione della finestra; appartiene al linguaggio delle forme generate dalla grammatica poiché non contiene punti etichettati.

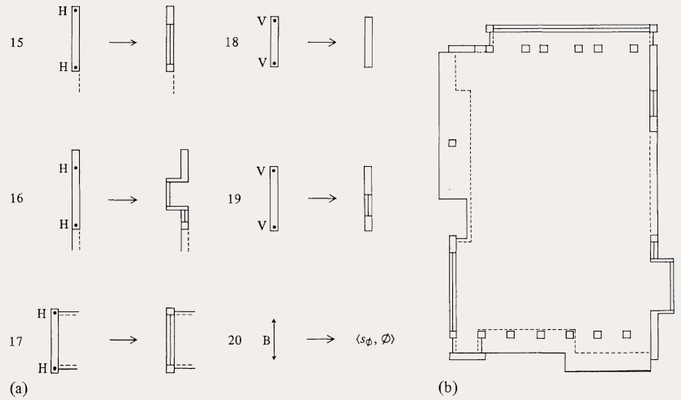


Figura **cs1.4**_7 (a) Schemi di progettazione di finestre e (b) forma S * generata dalla forma S5 dagli schemi di progettazione di finestre.

Realizzazione tridimensionale

Flemming indica un ulteriore sistema di regole che permettono la realizzazione del modello tridimensionale:

Le pareti verticali si elevano, in alzato, per tutta l'altezza dell'edificio, dal livello del suolo fino al solaio di copertura;

Le pareti orizzontali si elevano solo dal solaio di calpestio del primo piano al soffitto del terzo piano. Le finestre di tali muri devono avere davanzali sopra il livello del pavimento. Le nicchie sporgenti non possono toccare le nicchie corrispondenti ai piani superiori o inferiori.

Tutte le colonne devono raggiungere il livello del suolo e salire sopra il soffitto del terzo piano. Se l'alloggiamento tra due colonne è riempito da un muro verticale, entrambe le colonne devono estendersi oltre quel muro.

La figura cs1.4_8 mostra la realizzazione tridimensionale della forma S*. Il disegno mostra quegli elementi dei tre piani superiori della Casa Giuliani Frigerio che sembrano svolgere il ruolo più cruciale nella determinazione della forma complessiva.

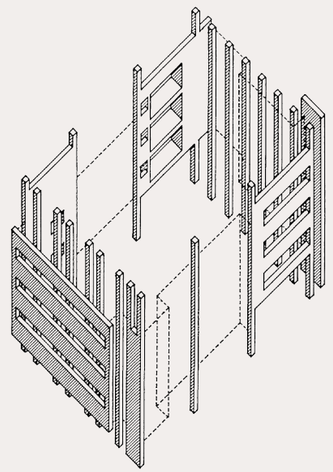


Figura **cs1.4**_8 La realizzazione tridimensionale della forma S*(vista nord-ovest).

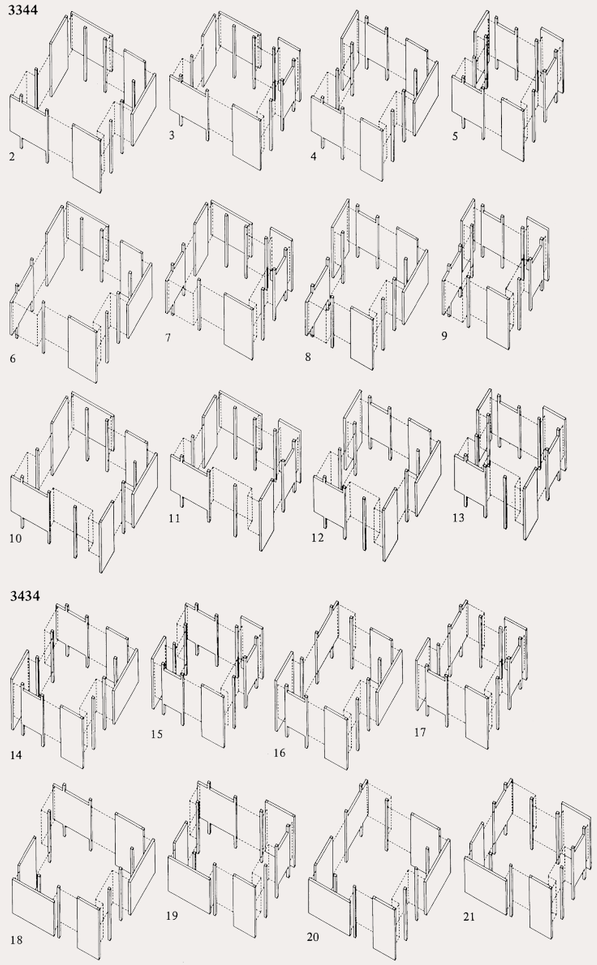


Figura cs1.4_9 Alcuni esempi delle classi di forme generate dalla grammatica.

cs1.5 Note conclusive

L'analisi di Eisenman del '71 si concentra dunque sull'ambiguità "pittorica" tra volume scavato e stratificazione di piani. Questa ambiguità si evidenzia nella contraddizione tra lettura frontale delle facciate e lettura obliqua. Gli angoli disgiunti infatti negano l'interpretazione volumetrica suggerita, in particolare, dagli aggetti delle facciate nord e est, pertanto Eisenman analizza le quattro facciate separatamente, considerandole sistemi autonomi.
Flemming, d'altro canto, lavorando sui singoli elementi in modo ricorsivo, identifica un pattern che sfugge a Eisenman: un'alternanza nell'articolazione dei setti *orizzontali* e *verticali* lungo le facciate (figura cs1.4_10). Non solo, il pattern è individuabile solo isolando i piani intermedi dell'edificio. Pertanto, la discontinuità apparente delle quattro facciate è contraddetta dal fatto che sia rintracciabile un pattern compositivo che le accomuna limitatamente ai tre piani intermedi. Le porzioni di facciate relative a questi piani costituiscono dunque un unico sistema governato da una precisa legge compositiva, che non emerge dalla lettura delle facciate considerate nella loro interezza. Inoltre, l'"emergenza" del pattern sembra essere conseguenza della metodologia utilizzata, che prevede l'addizione progressiva dei sistemi di elementi architettonici (colonne, muri, aperture) sull'intero perimetro dell'edificio.

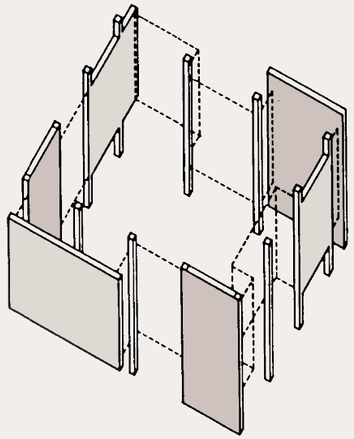
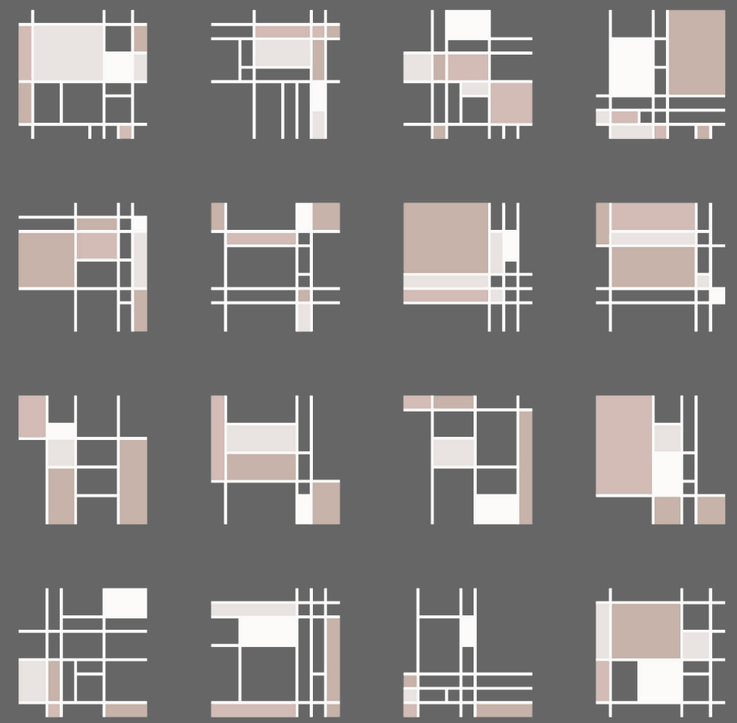


Figura **cs1.4**_10 Pattern alternato dei setti verticali e orizzontali secondo l'analisi di Flemming.

CASO STUDIO 2

Mondrian Grammar



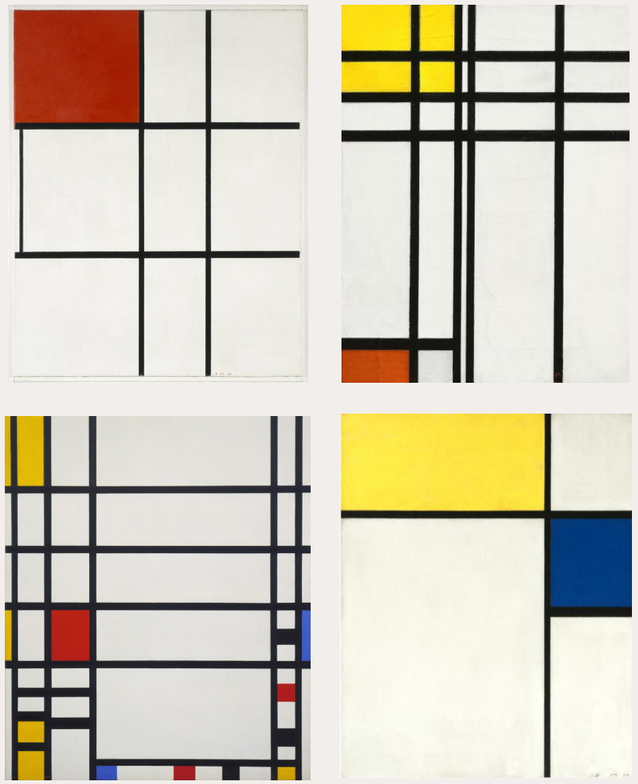


Figura cs2.1_1 Alcuni lavori di Mondrian (dall'alto in senso orario): Composition B (No.II) with Red, 1935, Tate (https://www.tate.org.uk); Opposition of Lines, Red and Yellow, 1937, Philadelphia Museum of Art (https://www.philamuseum.org); Trafalgar Square, 1939-43, Moma (https://www.moma.org); Composition with Blue and Yellow, 1932, Philadelphia Museum of Art (https://www.philamuseum.org).

Mondrian grammar

I dipinti di Mondrian hanno avuto una profonda influenza sull'architettura. De Stijl, dopotutto, stato è un movimento che, con van Doesburg come uno dei suoi pilastri, ha riunito pittori e architetti nella speranza di raggiungere un'idea di "creazione collettiva". Oltre agli architetti di De Stijl, molti altri architetti, da Hejduk a Mies, hanno chiaramente subito l'influenza di Mondrian. Va notato inoltre che furono gli architetti ad apprezzare particolarmente l'arte di Mondrian, formando de facto il più importante gruppo sociale dei suoi ammiratori, così come i suoi collezionisti più fedeli[1].
Non sorprende dunque vi siano stati diversi studi, anche nell'ambito del design computazionale, che hanno avuto come oggetto di ricerca le composizioni di questo artista. Shape grammar, ad esempio, è già stata utilizzata con successo come strumento analitico per descrivere la trasformazione degli stili di altri artisti neoplastici come Georges Vantongerloo e Fritz Glarner[2] e, più di recente, per analizzare l'opera dello stesso Mondrian[3].

In questa sezione proponiamo uno studio comparativo tra shape grammar e altri metodi computazionali utilizzati a questo scopo, che variano dagli algoritmi genetici alle reti neurali artificiali, al fine di comprendere vantaggi e limiti di ciascuno e, eventualmente, immaginare come una combinazione tra questi possa portare a risultati migliori o più interessanti. In particolare, prenderemo in esame i lavori di Schnier & Gero[4] e Lee[5], che hanno in comune il fatto di ibridare lo stile di Mondrian con opere di architetti (Wright nel primo caso, Hejduk nel secondo).

1 Bois, Y. (1987)"Mondrian and the Theory of Architecture," Assemblage 4, 102-130.
2 Knight, T. W. (1989). "Transformations of De Stijl Art: The Paintings of Georges Vantongerloo and Fritz Glarner". Environment and Planning B: Planning and Design, 16(1), 51-98. Vedi anche Knight, T. W. (1994). Transformations in design: a formal approach to stylistic change and innovation in the visual arts, Cambridge University Press.
3 Mondal, J. (2019). "Procedural Generation of Mondrian's Neo-plastic 2D Compositions for Stylistic Massing Extrusion using Shape Grammar", Proceedings of International Conference on Architecture Pedagogy (ICAP-2019) 11 - 13 November 2019, Faculty of Architecture & Ekistics, Jamia Millia Islamia, New Delhi, India.
4 Schnier, T., & Gero, J. S. (1998). "From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations", Adaptive Computing in Design and Manufacture, Springer, London, 207-219.
5 Lee, J. (1998). "Modelling Mondrian's Design Processes and Their Architectural Associations Using Multilayer Neural Networks", in T. Sasada, S. Yamaguchi, M. Morozumi, A. Kaga, and R. Homma (eds.), *CAADRIA '98 [Proceedings of The Third Conference on Computer Aided Architectural Design Research in Asia]* Osaka (Japan) 22-24 April 1998, 455-464.

Prima di procedere con lo studio comparativo, riportiamo, nel prossimo paragrafo, i passaggi per costruire una semplice shape grammar capace di restituire il metodo compositivo di Mondrian, così come riportati da Mondal[1].

1 Mondal, J. (2019). "Procedural Generation of Mondrian's Neo-plastic 2D Compositions for Stylistic Massing Extrusion using Shape Grammar".

cs2.1 Mondrian grammar

Dall'osservazione delle composizioni di Mondrian, alcune delle quali sono riportate in figura cs2.1_1, notiamo che normalmente si compongono dei seguenti elementi (figura cs2.1_2):

- Il rettangolo di base definito dal perimetro della tela;
- Un sistema di linee perpendicolari continue, che vanno da un bordo all'altro del rettangolo;
- Un sistema di segmenti posizionati tra le linee continue e tra queste e il rettangolo perimetrale;
- Un sistema di superfici colorate, selezionate tra le aree comprese tra le linee (consideriamo le aree bianche come "vuote").

Con un'osservazione più attenta è possibile notare una modularità nella distribuzione delle linee che rivela la presenza di una griglia soggiacente (figura cs2.1_3).

Figura cs2.1_2 Elementi della composizione del quadro "Composition with Large Blue Plane, Red, Black, Yellow, and Gray" (1921): a) composizione; b) rettangolo base; c) linee continue; d) segmenti; e) superfici. (Mondal, (2019). "Procedural Generation of Mondrian's Neo-plastic 2D Compositions for Stylistic Massing Extrusion using Shape Grammar")

Possiamo dunque individuare i passaggi necessari a impostare la grammatica:

- Passaggio 1: disegniamo il rettangolo di base,
- Passaggio 2: sovrapponiamo una griglia ortogonale al rettangolo,
- Passaggio 3: estrapoliamo dalla griglia alcune linee, di queste alcune resteranno continue, altre saranno divise in segmenti,
- Passaggio 4: selezioniamo alcune delle aree comprese tra le linee,
- Passaggio 5: differenziamo le aree con colori diversi.

La suddivisione del rettangolo con linee (passaggio 3) viene controllata utilizzando shape grammar come spiegato nelle seguenti sottosezioni.

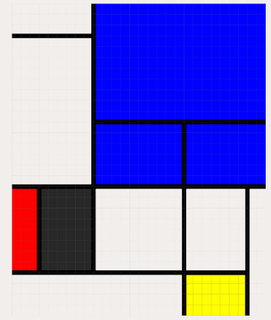


Figura cs2.1_3 Griglia modulare della composizione "Composition with Large Blue Plane, Red, Black, Yellow, and Gray" (1921)

Rule-set L'utilizzo della grammatica per la suddivisione sequenziale dei rettangoli rende il processo di natura operativa e di facile esecuzione mediante codifica, modellazione esplicita o disegno manuale. Le regole si applicano in sequenza. La forma iniziale è il rettangolo di base. Le regole A e B aggiungono le linee di suddivisione continue parallele rispettivamente all'asse x e all'asse y, e le regole C e D aggiungono linee di divisione secondarie (segmenti) parallele rispettivamente all'asse x e all'asse y (figura cs2.1_4).

- Regola A (linee continue parallele all'asse x): aggiunge una linea parallela all'asse x, selezionandola dalle linee orizzontali della griglia.
- Regola B (linee continue parallele all'asse y): aggiunge una linea parallela all'asse y, selezionandola dalle linee verticali della griglia.
- Regola C (linee di divisione parallele all'asse x): seleziona una delle linee orizzontali della griglia e la divide in base all'intersezione con le linee verticali ottenute con le regole B e D. Uno dei segmenti così ottenuti viene quindi aggiunto come linea di divisione.
- Regola D (linee di divisione parallele all'asse y): seleziona una delle linee verticali della griglia e la divide in base all'intersezione con le linee orizzontali ottenute con le regole A e C. Uno dei segmenti così ottenuti viene quindi aggiunto come linea di divisione.

Applicazione della grammatica La figura cs2.1_5 mostra varie suddivisioni sequenziali di un quadrato utilizzando AABBDC come stringa della grammatica. Diverse linee di suddivisione della griglia vengono aggiunte al quadrato con l'aggiunta di ciascuna regola dalla stringa. Successivamente, una parte delle aree comprese tra le linee viene selezionata per la colorazione.

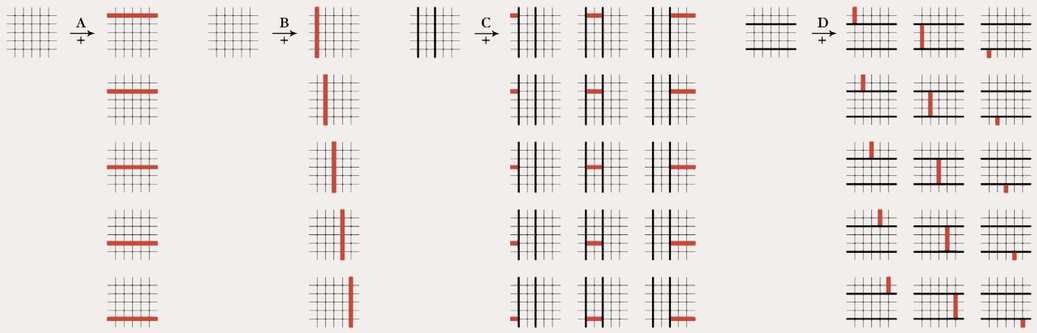


Figura cs2.1_4 Rule-set della grammatica (Mondal, (2019). "Procedural Generation of Mondrian's Neo-plastic 2D Compositions for Stylistic Massing Extrusion using Shape Grammar")

Le variabili dell'algoritmo sono le seguenti:

- Variabile 1: proporzioni della tela,
- Variabile 2: numero di linee della griglia nelle direzioni x e y,
- Variabile 3: stringa della grammatica,
- Variabile 4: ordine di selezione delle linee della griglia (linee di suddivisione) in base alla stringa,
- Variabile 5: numero di sottodivisioni da mantenere,
- Variabile 6: ordine di selezione delle sottodivisioni,
- Variabile 7: ordine di distribuzione dei quattro colori nelle suddivisioni selezionate.

I risultati della grammatica sono riportati in figura cs2.1_6.

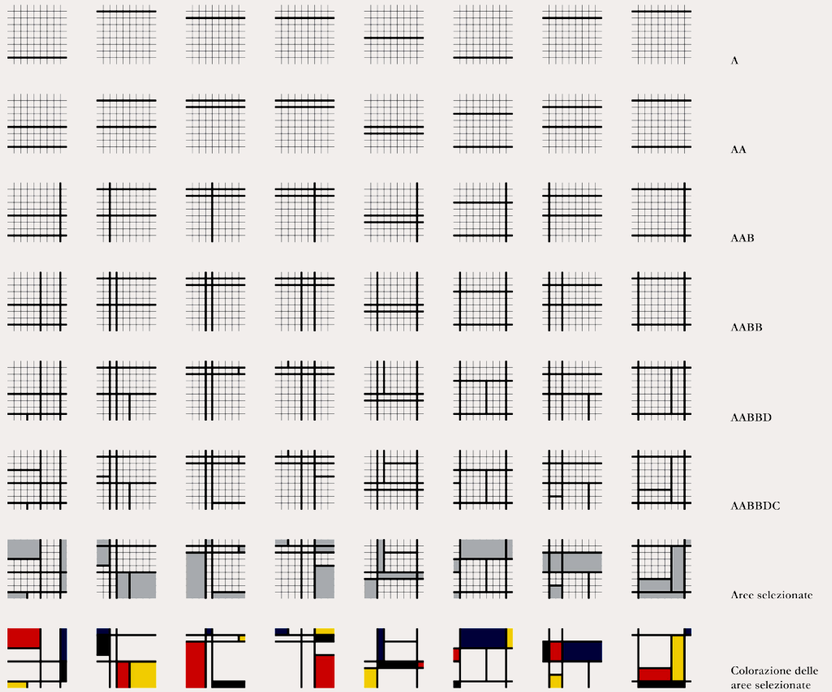


Figura cs2.1_5 Applicazione sequenziale delle regole (Mondal, (2019). "Procedural Generation of Mondrian's Neo-plastic 2D Compositions for Stylistic Massing Extrusion using Shape Grammar")

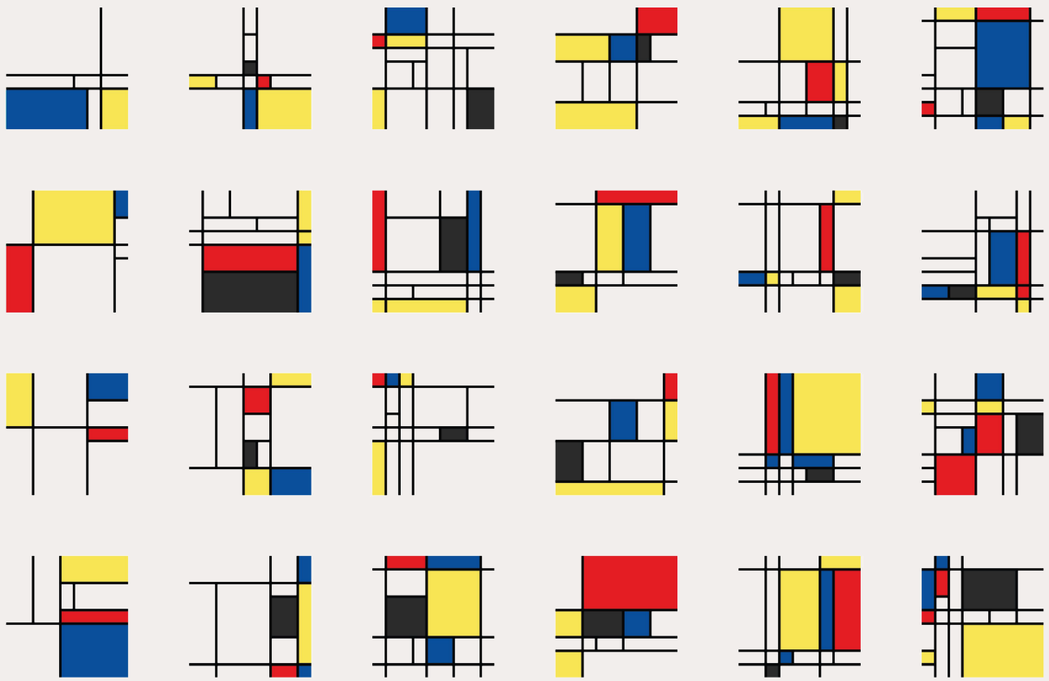


Figura cs2.1_6 Risultati con la Mondrian Grammar (Mondal, (2019). “Procedural Generation of Mondrian’s Neo-plastic 2D Compositions for Stylistic Massing Extrusion using Shape Grammar”)

cs2.2 Schnier e Gero: “From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations”

Nell’articolo “From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations”[1], Schnier e Gero usano un sistema evolutivo[2] per consentire al sistema di apprendere automaticamente lo stile di Mondrian, senza la necessità di una rappresentazione esplicita dello stile stesso, come invece avviene con shape grammar. In questo caso, per apprendere le caratteristiche dello stile, il sistema è programmato per provare a creare copie dell’esempio o degli esempi forniti; la funzione fitness è la misura della distanza tra il fenotipo corrente e l’esempio. Allo stesso tempo, le combinazioni di geni particolarmente efficaci nei genotipi vengono identificate e incapsulate in geni “evoluti”. Come risultato di questa “ingegneria genetica”, la rappresentazione si “evolve” e lo spazio di ricerca si trasforma in modo tale che la ricerca sia sempre più sbilanciata verso soluzioni simili agli esempi forniti. Il lavoro mostra inoltre come l’applicabilità di tale “rappresentazione evoluta” possa essere estesa mediante l’introduzione di trasformazioni che consentono di mescolare stili diversi, con un meccanismo simile all’incrocio di razze diverse nel regno animale, con l’ulteriore possibilità di controllare esattamente quali caratteristiche vengono utilizzate da quale fonte. Ciò può essere ottenuto mescolando diverse rappresentazioni apprese da diversi esempi e quindi utilizzando la nuova rappresentazione combinata per creare nuovi progetti. A titolo di esempio, Schnier e Gero mostrano come le informazioni sullo stile apprese da una serie di dipinti di Mondrian possano essere combinate con le informazioni sullo stile del design di una finestra della Hollyhock House di Frank Lloyd Wright, per creare nuovi progetti di finestre.

Rappresentazioni dell’ apprendimento

Per rappresentare i dipinti di Mondrian, viene utilizzata una codifica ad albero, in cui ogni nodo dell’albero corrisponde a una divisione di un rettangolo in due rettangoli più piccoli (figura cs2.2_1).

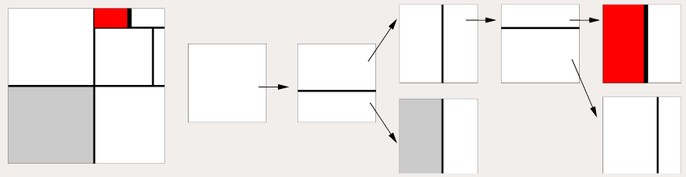


Figura cs2.2_1 Rappresentazione di un dipinto di Mondrian come serie di divisioni rettangolari strutturate ad albero. (Schnier & Gero (1998). “From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations”)

1 Schnier, T., & Gero, J. S. (1998). “From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations”, Adaptive Computing in Design and Manufacture, Springer, London, 207-219.

2 Per la descrizione degli algoritmi genetici vedi Appendice 2.

La figura cs2.2_2 mostra gli esempi utilizzati per generare una rappresentazione evoluta basata sui dipinti di Mondrian. Per rappresentare completamente un singolo dipinto è necessario un numero di nodi pari al numero di aree presenti nel dipinto stesso. Su ogni nodo, il sistema evolutivo può scegliere tra quattro diverse posizioni (in alto, in basso, a sinistra, a destra), quindici frazioni, quattro larghezze di linea e dodici colori.

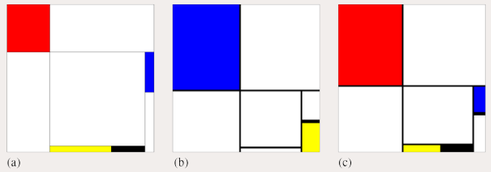


Figura cs2.2_2 Dipinti di Mondrian utilizzati per creare rappresentazioni evolute. (Schnier & Gero (1998). "From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations")

Per ciascuno dei tre esempi, cinque valori di fitness descrivono quanto il fenotipo sia "vicino" all'esempio in termini di posizione delle divisioni, correttezza dei colori e ampiezza delle linee, completezza e assenza di divisioni aggiuntive.
L'applicazione ha prodotto 110 geni evoluti, il primo e gli ultimi cinque geni creati sono mostrati nella figura cs2.2_3.

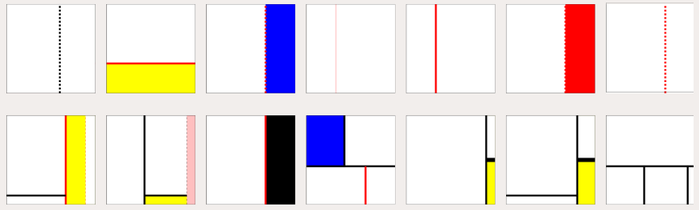


Figura cs2.2_3 Geni evoluti creati dagli esempi in Fig. cs2.2_2. (Schnier & Gero (1998). "From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations")

Da Mondrian a Frank Lloyd Wright

Come secondo esempio, è stato utilizzato il design di una finestra della Hollyhock House di Wright.
La figura cs2.2_4 (a) mostra il disegno utilizzato per generare la rappresentazione evoluta. Nell'originale, i rettangoli azzurri hanno una cornice più ampia, che è stata però modificata perché non poteva essere rappresentata nella rappresentazione di base. Per confrontare l'esempio con i fenotipi, il rettangolo esterno della finestra viene trasformato in un quadrato, figura cs2.2_4 (b).
A causa della maggiore complessità di questo esempio, il sistema ha appreso più geni evoluti, 159. Gli ultimi 11 sono mostrati nella figura cs2.2_4 (c).

Per utilizzare le rappresentazioni evolute, viene creato un insieme di individui iniziali utilizzando la rappresentazione, viene quindi eseguito un sistema evolutivo

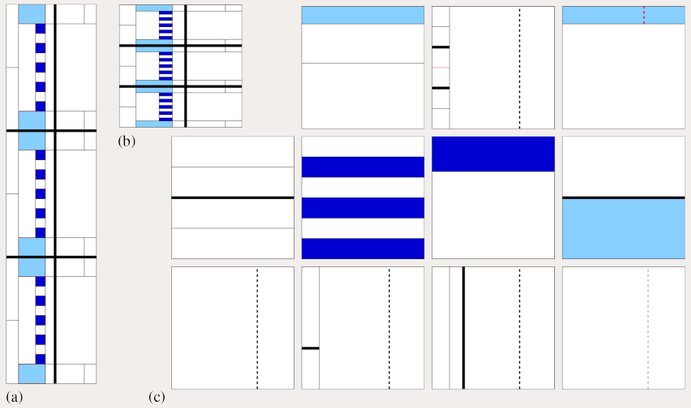


Figura cs2.2_4 Geni evoluti creati dagli esempi in Fig. cs2.2_2. (Schnier & Gero (1998). "From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations")

con una forma che descrive il nuovo design desiderato. Poiché le rappresentazioni evolute dei dipinti e il design della finestra utilizzano le stesse rappresentazioni di base, possono essere semplicemente combinate creando una popolazione iniziale casuale utilizzando geni evoluti di entrambi i set. La funzione fitness è volutamente scelta in modo da risultare neutra rispetto all'uso delle caratteristiche stilistiche, non le previene né le promuove.

La figura cs2.2_5 (a) mostra i nuovi progetti utilizzando solo la rappresentazione di base e le figure cs2.2_5 (b) e 6 (c) mostrano i risultati utilizzando le rappresentazioni create solo dai dipinti di Mondrian e solo dalla finestra di Frank Lloyd Wright. Le figure cs2.2_5 (d) ed (e) mostrano i risultati della combinazione delle rappresentazioni evolute create dai dipinti e dalla finestra in due modi diversi.
Per creare i risultati nelle figure cs2.2_5 (d), la popolazione iniziale è stata creata scegliendo casualmente geni evoluti da entrambe le fonti. Per i risultati nella Figura cs2.2_5 (e), le rappresentazioni evolute sono state modificate prima di essere utilizzate per creare individui iniziali: in tutti i geni evoluti creati dai dipinti di Mondrian, le informazioni sulla posizione e la frazione della divisione del rettangolo sono state rimosse. Allo stesso modo, in tutti i geni evoluti dal design della finestra, tutte le informazioni sul colore e lo spessore della linea sono state rimosse. I geni risultanti sono stati quindi utilizzati per creare gli individui iniziali. I risultati mostrano le caratteristiche topologiche ereditate dal design delle finestre di Frank Lloyd Wright e la colorazione e lo spessore delle linee dai dipinti di Mondrian.

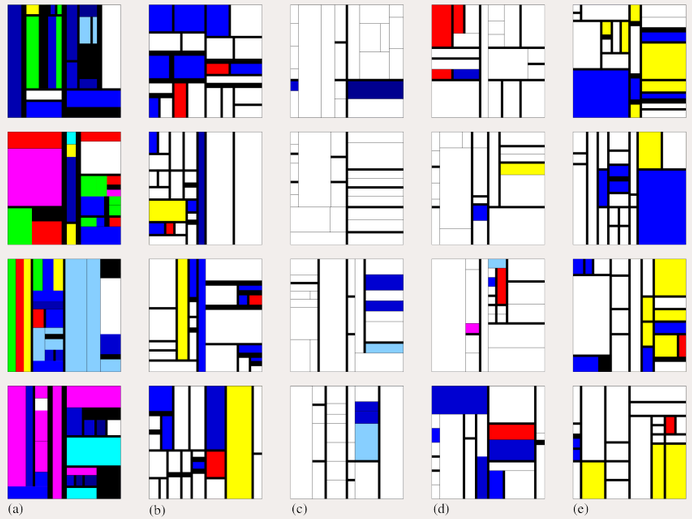


Figura cs2.2_5 Nuovi progetti: (a) senza rappresentazione evoluta, (b) utilizzando la rappresentazione evoluta dai dipinti di Mondrian, (c) utilizzando la rappresentazione evoluta dalla finestra di Frank Lloyd Wright, (d) utilizzando i geni completi di entrambe le rappresentazioni, (e) utilizzando la topologia dal design della finestra e informazioni sul colore e sullo spessore della linea dai dipinti. (Schnier & Gero (1998). "From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations")

cs2.3 Lee: "Modelling Mondrian's design processes and their architectural associations using multilayer neural networks".

Nell'articolo "Modelling Mondrian's design processes and their architectural associations using multilayer neural networks"[1], Lee propone uno studio sull'uso delle reti neurali nella progettazione il cui obiettivo è simulare la mappatura tra gli schizzi iniziali e gli esiti finali del progetto. Si vuole quindi simulare l'intero processo di progettazione, dalla fase iniziale a quella finale, utilizzando le reti neurali artificiali[2].

Addestramento e apprendimento

Lee seleziona per, l'addestramento della rete neurale, otto opere realizzate da Mondrian dal 1929 al 1932 mostrate in figura cs2.3_1. Le opere 1-7 costituiscono la "base di conoscenza", mentre l'ottava è usata per il test della rete. Per trasformare le opere in "0" e "1", accettabili per le reti neurali, queste sono state trasformate in bitmap 400X400. Queste costituiscono gli output della rete. Per quanto riguarda gli input, sono delle schematizzazioni delle composizioni che mettono in evidenza i rettangoli principali. Non vengono presi in considerazione né i colori, né lo spessore delle linee.

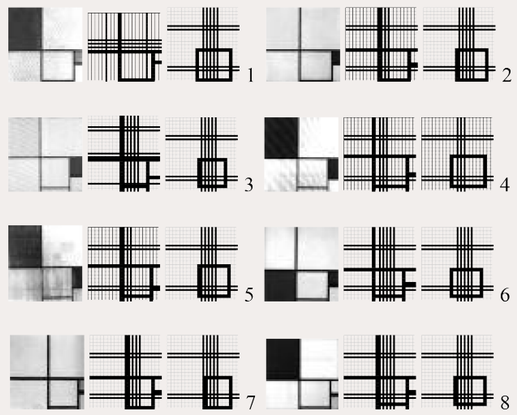


Figura cs2.3_1 Opere selezionate per l'addestramento della rete. I pattern1-7 sono i modelli di apprendimento per le reti neurali. Il pattern 8 è per il test. Per ciascuna opera sono riportate (da sinistra a destra): l'immagine dell'opera originale, la bitmap di output e la bitmap di input. (Lee (1998). "Modelling Mondrian's Design Processes and Their Architectural Associations Using Multilayer Neural Networks")

Una volta decise le unità di input e output, vengono impostate le variabili all'interno delle reti neurali, come il numero di livelli nascosti, la velocità di apprendimento ecc. In questa sperimentazione, sono stati impostati un livello nascosto, 100 unità

1 Lee, J. (1998). "Modelling Mondrian's Design Processes and Their Architectural Associations Using Multilayer Neural Networks", in T. Sasada, S. Yamaguchi, M. Morozumi, A. Kaga, and R. Homma (eds.), CAADRIA '98 [Proceedings of The Third Conference on Computer Aided Architectural Design Research in Asia] Osaka (Japan) 22-24 April 1998, 455-464.

2 Cfr. Appendice 3.

e una velocità di apprendimento di 0,001. Dopo 14319 cicli (o "epoche") di addestramento e apprendimento, il sistema risulta "ben addestrato".

Test

Per verificare che l'apprendimento sia avvenuto correttamente, si attua un primo test, come mostrato in figura cs2.3_2. Vengono immesse nella rete neurale le bitmap di input dei pattern 5 e 7, ottenendo un output che coincide con l'opera di Mondrian corrispondente.

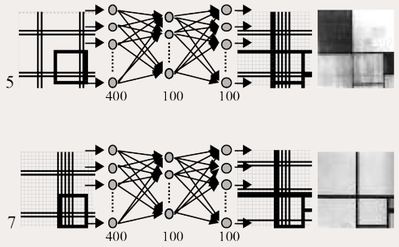


Figura cs2.3_2 Primo test dell'apprendimento. (Lee (1998). "Modelling Mondrian's Design Processes and Their Architectural Associations Using Multilayer Neural Networks")

Viene quindi eseguito un secondo test, inserendo nel sistema il pattern 8. Questo pattern, tuttavia, non è stato appreso dalle reti e l'output risulta incompleto. Il processo di input e output è mostrato in figura cs2.3_3.

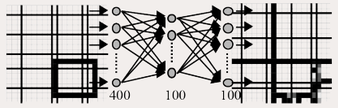


Figura cs2.3_3 Secondo test dell'apprendimento. A sinistra vi è la bitmap di input, a destra quella di output. Come si può notare dal "disturbo" sulla bitmap a destra, l'output è incompleto. (Lee (1998). "Modelling Mondrian's Design Processes and Their Architectural Associations Using Multilayer Neural Networks")

Per ovviare al problema viene impostato un valore di soglia (TRA, Threshold of Recognizing Activation), rispetto al quale viene testato il valore di ogni nodo: si ha 1 se il valore è maggiore del TRA e 0 se è minore. Secondo l'autore, il TRA riflette l'esperienza dei designer: i progettisti più esperti tenderanno a impostare valori TRA più bassi.

Vengono quindi sommati i valori dei bit incompleti in direzione verticale o orizzontale in una riga nelle bitmap generate dall'output. Tale somma è chiamata valore STV (straight total value) e determina la posizione della linea. Si procede quindi a confrontare il valore ottenuto con quello delle linee adiacenti. La linea con il valore STV maggiore determinerà la posizione della linea finale.

Ciò non è molto diverso da quanto accade normalmente durante il processo progettuale: negli schizzi iniziali di un progetto infatti, l'architetto torna più e più volte sul disegno per perfezionare la posizione di una semplice linea. Infine, trovata la posizione e l'organizzazione esatte, traccia la linea con maggiore forza, per farla risaltare rispetto agli altri segni di studio. Questa linea è la linea con il valore STV maggiore. In figura cs2.3_4 possiamo ritrovare questo processo nel paragone tra un quadro di Mondrian (non finito) e lo schizzo di studio.

Dopo l'introduzione del valore STV, l'output dell'immagine 5 ha prodotto la composizione di Mondrian che non era stato appreso dalle reti neurali (figura cs2.3_5).

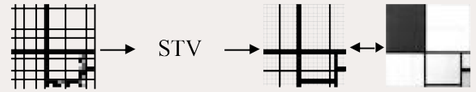


Figura cs2.3_5 Risultato del processo dopo l'introduzione del valore STV. (Lee (1998). "Modelling Mondrian's Design Processes and Their Architectural Associations Using Multilayer Neural Networks")

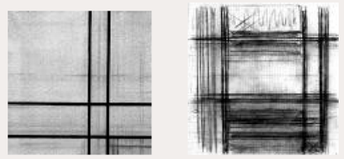


Figura cs2.3_4 A sinistra: "Composition with Yellow" (non finito). A destra: Composizione (studio). (Lee (1998). "Modelling Mondrian's Design Processes and Their Architectural Associations Using Multilayer Neural Networks")

Si effettua quindi un ultimo test, usando come input un'immagine che non è una composizione di Mondrian. Anche in questo caso, senza STV, l'output risulta incompleto, una volta inserito il valore STV invece si ottiene il risultato voluto (figura cs2.3_6). Il sistema ha prodotto dunque una composizione in "stile Mondrian".

Figura cs2.3_6 Composizione in stile Mondrian ottenuta con le reti neurali. (Lee (1998). "Modelling Mondrian's Design Processes and Their Architectural Associations Using Multilayer Neural Networks")

Da Mondrian a Hejduk

Data la forte influenza esercitata da Mondrian sull'architettura di John Hejduk, Lee propone un ulteriore esperimento: ibridare i due stili compositivi. Utilizza quindi per l'addestramento 3 opere di Mondrian (M1, M2, M3) e 3 planimetrie di Hejduk (H1, H2 e H3, rispettivamente. Texas House 7, Texas House 3 e Apartment House) e ripete la stessa procedura. I risultati sono mostrati in figura cs2.3_7. Per quanto l'esperimento possa considerarsi riuscito, lo stesso autore non ritiene l'esito particolarmente interessante e sottolinea la necessità di svolgere ulteriori ricerche a riguardo.

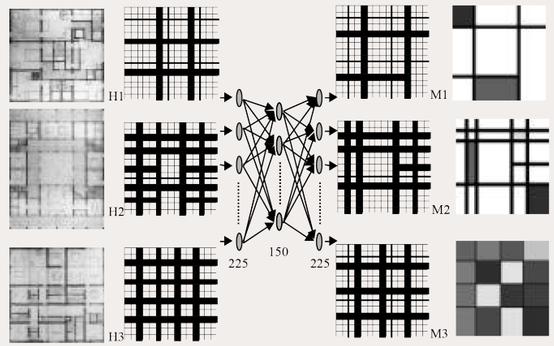


Figura cs2.3_7 MH neural networks system. (Lee (1998). "Modelling Mondrian's Design Processes and Their Architectural Associations Using Multilayer Neural Networks")

cs2.4 Osservazioni

I lavori illustrati nei paragrafi cs2.2 e cs2.3 utilizzano metodi alternativi a shape grammar per l'acquisizione dello stile di Mondrian. Già nel 1996, Schnier e Gero[1] avevano proposto un approccio di questo tipo, sfruttando l'idea che è possibile classificare lo stile a partire dalle sue caratteristiche visibili[2]. L'approccio crea una rappresentazione implicita delle caratteristiche dello stile, richiedendo solo una serie di modelli di esempio. Le conoscenze acquisite possono essere utilizzate direttamente per creare nuovi progetti che mostrano somiglianze di stile con i progetti di esempio, ma allo stesso tempo sono adattati a diverse condizioni di progettazione[3]. I risultati di questi lavori ci sembrano molto interessanti, anche se, in entrambi i casi, meno coerenti con gli esempi forniti al sistema per il training di quanto non siano quelli ottenuti con shape grammar. Bisogna tenere in considerazione che entrambi i lavori risalgono agli anni Novanta e che oggi gli avanzamenti nella ricerca permetterebbero di ottenere risultati migliori. La scelta di questi esempi è stata dettata principalmente dal fatto che erano immediatamente comparabili con la semplice grammatica illustrata nel primo paragrafo.

Nel 2010 Andrzejewski et al. hanno proposto un metodo di analisi dove modelli generativi e classificatori di alberi decisionali vengono addestrati con esempi positivi e negativi delle opere di Mondrian[4].

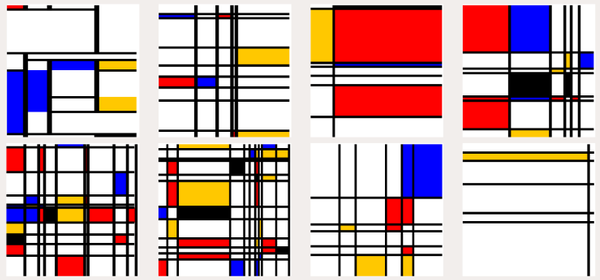


Figura cs2.4_1 Risultati della sperimentazione di Andrzejewski et al. (Andrzejewski et al. (2010). "Inferring compositional style in the neo-plastic paintings of Piet Mondrian by machine learning".)

1 Schnier, T. & Gero, J. S. (1996). Learning genetic representations as alternative to hand-coded shape grammars, in J. S. Gero & F. Sudweeks (eds), Arti_cial Intelligence in Design '96, Kluwer, Dordrecht, 39-57.

2 Chan, C.-S. (1995). A cognitive theory of style, Environment and Planning B: Planning and Design 22: 461.47.

3 Schnier, T., & Gero, J. S. (1998). "From Mondrian to Frank Lloyd Wright: Transforming evolving representations", Adaptive Computing in Design and Manufacture, Springer, London, 207-219.

4 Andrzejewski, D., Stork, D. G., Zhu, X., & Spronk, R. (2010, February). Inferring compositional style in the neo-plastic paintings of Piet Mondrian by machine learning. In Computer Vision and Image Analysis of Art, Vol. 7531, 75310G. International Society for Optics and Photonics.

I risultati (figura cs2.4_1), anche in questo caso, presentano delle incongruenze rispetto ai modelli di addestramento: hanno un numero maggiore di linee rispetto alla maggior parte delle opere nel set di addestramento, e un numero molto inferiore di linee non estese, cioè linee che terminano su linee perpendicolari piuttosto che sul bordo esterno del dipinto[1].

Naturalmente, la grammatica illustrata nel primo paragrafo è molto semplice e non tiene conto di molte questioni, prima tra tutte il sistema di proporzioni della composizione. Questo avrebbe reso la grammatica molto più articolata e difficile da costruire. Recentemente Park[2] ha proposto un metodo per generare riproduzioni stilistiche della "Composition with Red, Yellow, and Blue" di Mondrian (cs2.4_2), che utilizza shape grammar per l'analisi e gli algoritmi genetici per ottimizzare il sistema di proporzioni della composizione. I risultati sono molto accurati, come possiamo vedere dalla figura cs2.4_3, e ci portano a credere che allo stato attuale, per operazioni volte a dedurre la struttura compositiva di un certo stile o corpus di opere, shape grammar possa ancora considerarsi un ingrediente fondamentale.

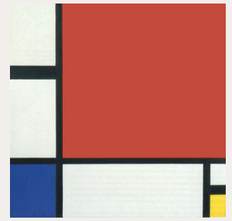


Figura cs2.4_2 Piet Mondrian, "Composition with Red, Yellow, and Blue", 1930 (Park (2020). "Stylistic reproductions of Mondrian's composition with red, yellow, and blue")

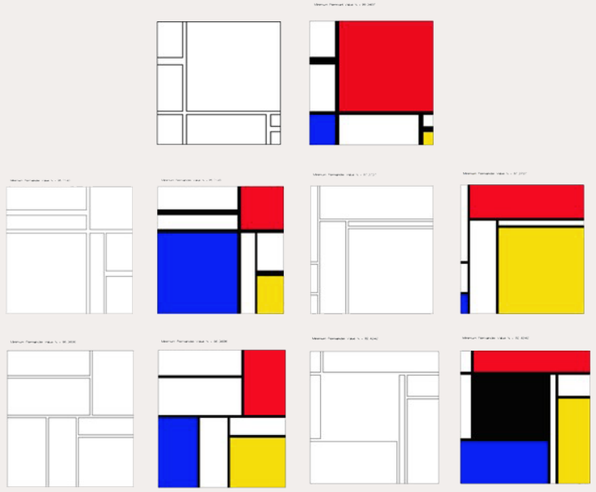


Figura cs2.4_3 Risultati della sperimentazione di Park (Park (2020). "Stylistic reproductions of Mondrian's composition with red, yellow, and blue")

1 Ibid.

2 Park, H-J. (2020). "Stylistic reproductions of Mondrian's composition with red, yellow, and blue", in D. Holzer, W. Nakapan, A. Globa, I. Koh (eds.), *RE: Anthropocene, Design in the Age of Humans - Proceedings of the 25th CAADRIA Conference - Volume 2, Chulalongkorn University, Bangkok, Thailand, 5-6 August 2020*, 133-142.

CAPITOLO 6

CONCLUSIONI

Alla luce di quanto detto, possiamo dire che shape grammar, nella sua forma tradizionale "esperta", è fondamentalmente un metodo per dedurre le regole compositive di un determinato stile o di un'opera architettonica, sia che venga utilizzata per analizzare tale stile o opera (capitolo 3), sia che ci si ponga l'obiettivo di sviluppare un nuovo progetto (capitolo 4). Per quanto sia possibile creare una grammatica "da zero", infatti, i risultati migliori si ottengono laddove shape grammar viene impiegata per un'indagine "critica". In altre parole, per *esplorare* un certo *spazio concettuale* e, eventualmente, *trasformarlo* (cfr. capitolo 1, paragrafo 1.4).
La costruzione di una grammatica a partire da uno stile dato rappresenta dunque un processo di apprendimento, e questa caratteristica, del resto, attribuisce a shape grammar un grande potenziale educativo. Secondo Knight infatti:

"There is no better way to learn about styles or languages of designs (at least compositionally) than by either studying shape grammars already written for languages or by writing grammars oneself. Good analytic grammars are both parsimonious and descriptive. They are eye-openers, revealing simplicity or regularities behind designs seemingly complex or random. They reveal the thoughtfulness, the "individual genius", behind designs that students might otherwise take as unfathomable"[1].

I requisiti che una grammatica analitica deve soddisfare, così come espressi da da Stiny e Mitchell[2], richiedono infatti uno sforzo di astrazione da parte dell'autore che gli permette di appropriarsi completamente della struttura logica del progetto esaminato:

"(1) it should clarify the underlying commonality of structure and appearance manifest for the buildings in the corpus; (2) it should supply the conventions and criteria necessary to determine whether any other building not in the original corpus is an instance of the

1 Knight, T. (1999). "Applications in architectural design, and education and practice". Report for the NSF. In *MIT Workshop on Shape Computation*, 1-11: 4. Traduzione: "Non c'è modo migliore per conoscere gli stili o i linguaggi di progettazione (almeno dal punto di vista compositivo) che studiare le shape grammars già scritte per tali linguaggi o scrivere personalmente le grammatiche. Le buone grammatiche analitiche sono sia parsimoniose che descrittive. Aprono gli occhi, rivelano semplicità o regolarità dietro progetti apparentemente complessi o casuali. Rivelano il pensiero, il "genio individuale", dietro i progetti che gli studenti potrebbero altrimenti considerare insondabili".
2 Stiny, G., & Mitchell, W. J. (1978). The palladian grammar. *Environment and planning B: Planning and design*, *5*(1), 5-18.

style; and (3) it should provide the compositional machinery needed to design new buildings that are instances of the style"[1].

Tale sforzo di astrazione, inoltre, rivela anche strategie generali di progettazione comuni a più grammatiche che possono essere utilizzate anche in lavori originali. Grammatiche diverse, create per linguaggi molto diversi tra loro, utilizzano spesso, infatti, strategie di progettazione comuni. Ad esempio, diverse grammatiche utilizzano la suddivisione come base per il processo generativo. La ice-ray grammar[2], la grammatica della sedia Hepplewhite[3] e la grammatica di Malagueira[4] funzionano tutte in questo modo. Altre grammatiche utilizzano un processo additivo per la generazione di progetti. Con questo approccio, le soluzioni vengono generate a partire dal "nucleo" del progetto a cui vengono successivamente aggiunte altre parti. Viene utilizzato questo processo, ad esempio, nella grammatica di Wright[5], nella grammatica Queen Anne[6] e nella grammatica di Colakoglu[7]. Grammatiche architettoniche tridimensionali descrivono strategie per definire le connessioni verticali tra spazi. Ad esempio, nella grammatica Wright, nella grammatica Queen Anne e nella grammatica Siza, le organizzazioni spaziali del primo e del secondo piano sono fortemente connesse: l'organizzazione del secondo piano imita quella del primo e deriva da essa[8].

Tali strategie generali di progettazione che accomunano grammatiche diverse, così come quelle più specifiche incorporate in grammatiche particolari, sono strumenti critici e possono non corrispondere alla verità storica. Come abbiamo visto nel capitolo 3 e nel primo caso studio, la grammatica potrebbe non avere nulla a che fare con il modo in cui l'oggetto dell'analisi è stato concepito in origine. L'analisi è quindi intesa come uno strumento critico d'indagine che presuppone un'interpretazione da parte del suo autore.

Ad esempio, nel caso studio di Casa Giuliani Frigerio, abbiamo visto come il modo in cui Flemming imposta la grammatica porta al riconoscimento del pattern di setti orizzontali/verticali, che probabilmente non era assolutamente espressione di una scelta consapevole da parte di Terragni.

In questo senso, anche quando usata solo per analizzare opere esistenti, la grammatica ha già un potenziale creativo in quanto ha già parzialmente trasformato lo spazio di esplorazione. Ciò è ancor più vero quando abbiamo un'applicazione combinata di analisi/sintesi (come nel caso della grammatica di Malagueira). Del resto, pur partendo da un esempio esistente, la grammatica può essere modificata fino a perdere qualunque somiglianza col riferimento iniziale, seguendo un processo che va dall'apprendimento all'emancipazione.

Tuttavia, come abbiamo detto, è perfettamente possibile creare grammatiche da zero, ma ciò comporta il doversi confrontare con alcuni problemi. Senza un riferimento iniziale, infatti, la grammatica sarà tendenzialmente molto semplice e, come abbiamo visto nei capitolo 1 e 5, questo può portare alla generazione di molti risultati inutili. Un modo per ovviare a tale problema è sviluppare una grammatica che soddisfi gli obiettivi e i vincoli del progetto a cui si sta lavorando. In altre parole, a un certo punto del processo di sviluppo di una grammatica, se non all'inizio, è necessario stabilire una connessione tra le regole che descrivono la forma spaziale e gli obiettivi del progetto, altrimenti la grammatica potrebbe portare in qualunque direzione. Creare questa connessione non è un compito facile perché, come abbiamo visto nel capitolo 2, una delle caratteristiche fondamentali di shape grammar è che permette l'emergenza, cosa che la rende molto potente come strumento generativo, ma anche "imprevedibile". Regole apparentemente semplici possono produrre risultati sorprendentemente complessi.

Sono stati suggeriti diversi approcci per collegare grammatiche e obiettivi. Un approccio è diretto e implica la scrittura di regole i cui comportamenti ed esiti siano prevedibili in qualche modo. Questo però andrebbe a compromettere la creatività del progetto, limitando troppo la potenza generativa di shape grammar e rendendo il processo completamente deterministico.

Il problema dell'imprevedibilità era già stato sottolineato da Flemming in un articolo del 1987, dove fa una proposta:

"There is, at the present time, no body of theory available that would

1 Ibid., 17. Traduzione: "(1) dovrebbe chiarire la sottostante comunanza di struttura e aspetto visibile negli edifici del corpus; (2) dovrebbe fornire le convenzioni e i criteri necessari per determinare se qualsiasi altro edificio che non appartiene al corpus originale è un'istanza dello stile; e (3) dovrebbe fornire il meccanismo compositivo necessario per progettare nuovi edifici che sono istanze dello stile".

2 Stiny, G. (1977). "Ice-ray: a note on Chinese lattice designs," *Environment and Planning B* 4, 89-98.

3 Knight, T. W. (1980). The generation of Hepplewhite-style chair-back designs. *Environment and planning B: planning and design*, *7*(2), 227-238.

4 Duarte, J. P. (2001). *Customizing mass housing: a discursive grammar for Siza's Malagueira houses* (Doctoral dissertation, Massachusetts Institute of Technology).

5 Koning, H., & Eizenberg, J. (1981). The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and planning B: planning and design*, *8*(3), 295-323.

6 Flemming, U. (1987). More than the sum of parts: the grammar of Queen Anne houses. *Environment and Planning B: Planning and Design*, *14*(3), 323-350.

7 Colakoglu, B. (2005). Design by grammar: an interpretation and generation of vernacular hayat houses in contemporary context. *Environment and Planning B: Planning and Design*, *32*(1), 141-149.

8 Knight, T. (1999). "Applications in architectural design, and education and practice".

allow us to predict the properties of shapes generated by a grammar solely from an inspection of its rules. In order to assure that a grammar is properly constructed, we often have to enumerate a substantial number, if not all of the shapes it generates. This process is tedious and error-prone if done manually and could clearly gain from automation"[1].

Con questo approccio alternativo, le grammatiche vengono sviluppate senza una chiara idea dei loro risultati. Una strategia automatizzata di ricerca e test viene quindi utilizzata per esplorare lo spazio dei progetti generati, campionandoli e testandoli per vedere se soddisfano determinati obiettivi. Questo approccio, che abbiamo illustrato nel capitolo 5, prevede l'uso di tecniche di apprendimento automatico.
L'integrazione di grammatiche di forma semplici ("ingenue"), facili da elaborare e comprendere, con meccanismi di controllo che ne orientano il processo generativo, permette di sfruttare appieno la potenza creativa della grammatica riuscendo al contempo a soddisfare i requisiti di progettazione. Nel secondo caso studio, ad esempio, abbiamo visto come l'uso di shape grammar in combinazione con tecniche di machine learning sia risultato il più efficace e il più preciso, tra i metodi analizzati, nel replicare le composizioni di Mondrian.
Uno dei modi più interessanti in cui tali meccanismi di controllo possono guidare il processo generativo è scegliendo il modo in cui eseguire le regole, come illustrato nel lavoro di Ruiz-Montiel et al.[2] riportato nella scheda 8 (capitolo 5). Come abbiamo visto, infatti, uno dei vantaggi di shape grammar rispetto ad altri sistemi generativi risiede nella sua natura non deterministica. Pertanto, gli utenti delle grammatiche hanno la possibilità di scegliere i modi per eseguire le regole. In ogni fase della generazione di un progetto, queste scelte possono includere: la regola da applicare, dove applicare la regola, sotto quale trasformazione applicare la regola e l'assegnazione di valori ai parametri della regola. Queste scelte sono strutturate in una sequenza che può o meno riflettere il processo progettuale[1].
La maggior parte delle grammatiche analitiche tradizionali, ad esempio, non sono strutturate in questo modo. In altre parole, le scelte relative all'applicazione delle regole può non essere sensato da un punto di vista progettuale. Ciò è particolarmente vero per le scelte relative ai parametri. In alcune grammatiche, ad esempio, l'utente deve decidere le dimensioni dei singoli spazi in pianta prima che venga decisa la loro disposizione generale. Nella prima fase della grammatica palladiana, ad esempio, gli spazi vengono generati uno o due alla volta per definire la griglia sottostante della pianta. Ogni volta che uno spazio viene aggiunto, deve anche essere dimensionato. Pertanto, l'utente deve decidere le dimensioni e le proporzioni dei singoli spazi prima di conoscere (generare) le dimensioni e le proporzioni della griglia complessiva. Nella grammatica di Wright, le dimensioni e le proporzioni di alcuni spazi nell'unità centrale di una casa vengono decise prima che le funzioni di questi spazi vengano assegnate[2].
Le grammatiche "progettuali", invece, devono essere strutturate in modo da essere coerenti col processo progettuale. Un modo in cui è stato affrontato il problema, nelle grammatiche "esperte", vede l'uso di grammatiche "parallele", si usano cioè grammatiche diverse per generare diverse fasi di un progetto. È il caso della grammatica di Malgueira, dove Duarte le utilizza non solo come soluzione al problema dei parametri, ma come un modo per generare rappresentazioni multiple del progetto. Tuttavia, la soluzione di Ruiz-Montiel et al. sembra molto vantaggiosa, vista la complessità delle grammatiche parallele e la difficoltà che si possono riscontrare nel generarle.
In generale, l'integrazione di shape grammar con tecniche di machine learning sembra, in questo momento, un approccio molto promettente, che riesce a sfruttare al meglio la capacità generativa di shape grammar, risolvendo i problemi che di solito si riscontrano nell'uso dei sistemi esperti.

1 Flemming, U. (1987) "More than the sum of its parts: the grammar of Queen Anne houses," *Environment and Planning B: Planning and Design* 14, 349. Traduzione: "Non esiste, al momento, alcun corpo di teoria disponibile che ci consenta di prevedere le proprietà delle forme generate da una grammatica unicamente da un'ispezione delle sue regole. Per garantire che una grammatica sia costruita correttamente, spesso dobbiamo enumerare un numero sostanziale, se non tutte le forme che genera. Questo processo è noioso e soggetto a errori se eseguito manualmente e potrebbe chiaramente trarre vantaggio dall'automazione ".

2 Ruiz-Montiel, M., Boned, J., Gavilanes, J., Jiménez, E., Mandow, L., & PéRez-De-La-Cruz, J. L. (2013). Design with shape grammars and reinforcement learning. *Advanced Engineering Informatics*, *27*(2), 230-245.

1 Knight, T. (1999). "Applications in architectural design, and education and practice".

2 Ibid.

Appendice 1*: Machine learning e deep learning*

L'apprendimento automatico ("Machine Learning", ML) è sottoinsieme dell'intelligenza artificiale che studia algoritmi informatici capaci di apprendere e migliorare automaticamente attraverso l'esperienza[1]. Gli algoritmi di apprendimento automatico costruiscono un modello basato su dati di esempio, noti come "dati di addestramento", al fine di effettuare previsioni o decisioni senza essere esplicitamente programmati per farlo[2]. La programmazione convenzionale infatti prevede la codifica di un algoritmo, ossia un procedimento finito di regole e di operazioni, in un linguaggio di programmazione. Il programma realizzato sarà quindi in grado di eseguire solo le operazioni per le quali è stato ideato, quelle definite nell'algoritmo. L'apprendimento automatico ci consente di affrontare compiti troppo difficili da risolvere con programmi di questo tipo:

"The difficulties faced by systems relying on hard-coded knowledge suggests that AI systems need the ability to acquire their own knowledge, by extracting patterns from raw data. This capability is known as machine learning. The introduction of machine learning enabled computers to tackle problems involving knowledge of the real world and make decisions that appear subjective"[3].

Gli approcci all'apprendimento automatico sono tradizionalmente suddivisi in due grandi categorie, a seconda di come il sistema "sperimenta" un dataset, ossia un insieme di esempi:

apprendimento supervisionato: a ogni esempio del dataset è associata un'etichetta (label) o un obiettivo (target) forniti da un "insegnante" che mostra al sistema cosa fare;

apprendimento non supervisionato: non c'è istruttore o insegnante e l'algoritmo

1 Mitchell, T. (1997). Machine Learning. New York: McGraw Hill

2 La definizione "without being explicitly programmed" è spesso attribuita a Arthur Samuel, che ha coniato il termine "machine learning" nel 1959, ma la frase non si trova alla lettera in questa pubblicazione e potrebbe essere una parafrasi apparsa in seguito. Cfr. "Paraphrasing Arthur Samuel (1959), the question is: How can computers learn to solve problems without being explicitly programmed?" in Koza, John R.; Bennett, Forrest H.; Andre, David; Keane, Martin A. (1996). Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. Artificial Intelligence in Design '96. Springer, Dordrecht, 151–170.

3 Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning, MIT Press, 2-3.
Traduzione: "Le difficoltà incontrate dai sistemi che si basano su conoscenze hard-coded suggeriscono che i sistemi di intelligenza artificiale necessitano della capacità di acquisire la propria conoscenza, estraendo modelli da dati grezzi. Questa funzionalità è nota come apprendimento automatico. L'introduzione dell'apprendimento automatico ha consentito ai computer di affrontare i problemi che coinvolgono la conoscenza del mondo reale e di prendere decisioni che sembrano soggettive".

deve imparare a dare un senso ai dati senza questa guida[1].

Sebbene l'apprendimento senza supervisione e l'apprendimento supervisionato non siano concetti completamente formali o distinti, aiutano a classificare approssimativamente gli algoritmi di apprendimento automatico. Tradizionalmente, le persone si riferiscono a problemi di regressione e classificazione come apprendimento supervisionato. Nella classificazione si chiede al programma di specificare a quale delle k categorie appartiene un input; mentre nella regressione, dato un input, viene chiesto di prevedere un valore numerico. L'apprendimento non supervisionato invece è solitamente associato alla stima della densità o al clustering[2].

Sono possibili altre varianti del paradigma di apprendimento. Ad esempio, nell'apprendimento semi supervisionato, alcuni esempi includono un obiettivo di supervisione ma altri no. Alcuni algoritmi di apprendimento automatico non sperimentano solo un set di dati fisso. Ad esempio, gli algoritmi di apprendimento per rinforzo interagiscono con un ambiente, quindi esiste un ciclo di feedback tra il sistema di apprendimento e le sue esperienze[3].

Gli algoritmi di apprendimento automatico funzionano bene su un'ampia varietà di problemi importanti. Tuttavia, non sono riusciti a risolvere alcuni problemi centrali dell'IA, come il riconoscimento delle parole o il riconoscimento di oggetti. Lo sviluppo del deep learning è stato motivato in parte dall'incapacità degli algoritmi tradizionali di generalizzare bene su tali obiettivi dell'intelligenza artificiale[4].

Deep Learning

L'apprendimento profondo (noto anche come apprendimento strutturato profondo) fa parte di una più ampia famiglia di metodi di apprendimento automatico basati su reti neurali artificiali con representation learning[5]. Nell'apprendimento automatico, il representation learning è un insieme di tecniche che consente a un sistema di scoprire automaticamente le rappresentazioni necessarie per il rilevamento delle caratteristiche o la classificazione da dati grezzi[6]. La maggior parte dei

1 Goodfellow, et al. (2016). Deep Learning, capitolo 5.
2 Ibid.
3 Ibid.
4 Goodfellow, et al. (2016). Deep Learning.
5 Ibid.
6 Y. Bengio; A. Courville; P. Vincent (2013). "Representation Learning: A Review and New Perspectives". IEEE Transactions on Pattern Analysis and Machine Intelligence. 35 (8): 1798–1828; Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". Neural Networks. 61: 85–117; Bengio, Yoshua; LeCun, Yann; Hinton, Geoffrey (2015). "Deep Learning". Nature. 521 (7553): 436–444.

moderni modelli di deep learning si basa su reti neurali profonde, in particolare, Convolutional Neural Networks (CNN)[1].

La parola "deep" in "deep learning" si riferisce al numero di livelli attraverso i quali i dati vengono trasformati. Più precisamente, i sistemi di deep learning hanno una profondità sostanziale del credit assignment path (CAP)[2]. Il CAP è la catena di trasformazioni dall'input all'output che descrive le connessioni potenzialmente causali tra questi[3]. Il deep learning utilizza infatti più livelli per estrarre progressivamente funzionalità di livello superiore dall'input grezzo[4]. Ad esempio, nell'elaborazione delle immagini, i livelli inferiori possono identificare i bordi, mentre i livelli superiori possono identificare i concetti rilevanti per un essere umano come cifre, lettere o volti[5].

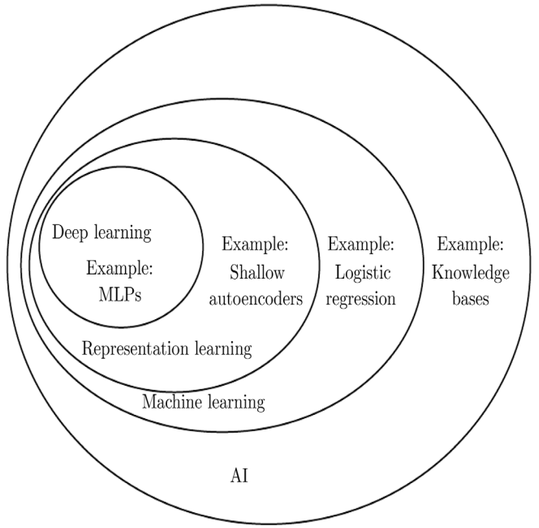


Figura A1_1 Un diagramma di Venn che mostra come l'apprendimento profondo sia una sorta di apprendimento della rappresentazione, che a sua volta è una sorta di apprendimento automatico, che viene utilizzato per molti ma non tutti gli approcci all'IA. Ogni sezione del diagramma di Venn include un esempio di tecnologia AI. (Goodfellow, et al. (2016). Deep Learning, MIT Press)

1 Goodfellow, et al. (2016). Deep Learning.
2 Ibid.
3 Ibid.
4 Deng, L., & Yu, D. (2014). Deep learning: methods and applications. ***Foundations and trends in signal processing***, ***7***(3–4), 197-387: 199-200.
5 Goodfellow, et al. (2016). Deep Learning.

Reti Neurali

Le reti neurali sono dunque la componente fondamentale degli algoritmi di Deep Learning. I primi modelli computazionali erano ad ispirazione biologica, cioè cercavano di mimare il cervello umano e le sinapsi[1], ma col tempo l'analogia si è persa: "Some of the earliest learning algorithms we recognize today were intended to be computational models of biological learning, i.e. models of how learning happens or could happen in the brain. As a result, one of the names that deep learning has gone by is artificial neural networks (ANNs). (...) The modern term "deep learning" goes beyond the neuroscientific perspective on the current breed of machine learning models. It appeals to a more general principle of learning multiple levels of composition, which can be applied in machine learning frameworks that are not necessarily neurally inspired"[2].

Modelli lineari

I primi predecessori del moderno deep learning erano semplici modelli lineari motivati da una prospettiva neuroscientifica. Questi modelli sono stati progettati per prendere un insieme di n valori di input $x_1 \ldots x_n$ e associarli a un output y. Apprendono quindi una serie di pesi $w_1 \ldots w_n$ e calcola il loro output:
$f(x, w) = x_1 w_1 + \cdots + x_n w_n$[3].

Il McCulloch-Pitts Neuron[4] è stato uno dei primi modelli di funzione cerebrale. Questo modello lineare potrebbe riconoscere due diverse categorie di input verificando se f (x, w) è positivo o negativo[5]. Ovviamente, affinché il modello corrispondesse alla definizione desiderata delle categorie, i pesi dovevano essere impostati correttamente[6]. Questi pesi possono essere impostati dall'operatore umano. Negli anni '50, il perceptron[7] è diventato il primo modello in grado di apprendere i pesi che definiscono le categorie fornite come esempi di input da ciascuna categoria[1]. L'Adaptive Linear Element (ADALINE), che risale all'incirca nello stesso periodo, ha semplicemente restituito il valore di f(x) stesso per prevedere un numero reale[2] e potrebbe anche imparare a prevedere questi numeri da dati[3].

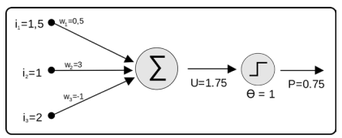


Figura A1_2 Neurone artificiale proposto da McCulloch e Pitts (https://it.wikipedia.org/wiki/Rete_neurale_artificiale).

Questi semplici algoritmi di apprendimento hanno influenzato notevolmente il panorama moderno dell'apprendimento automatico. L'algoritmo di addestramento utilizzato per adattare i pesi dell'ADALINE era un caso speciale di un algoritmo chiamato discesa del gradiente stocastico (*stochastic gradient descent)*[4]. Versioni leggermente modificate di questo algoritmo rimangono ancora oggi gli algoritmi di addestramento dominanti per i modelli di apprendimento profondo[5].

I modelli basati su f (x, w) usati dal perceptron e da ADALINE sono chiamati modelli lineari[6]. Questi modelli rimangono alcuni dei modelli di apprendimento automatico più utilizzati, anche se in molti casi vengono addestrati in modi diversi rispetto ai modelli originali[7].

I modelli lineari hanno molte limitazioni. La più famosa è che non possono apprendere la funzione XOR, dove f ([0, 1], w) = 1 e f ([1, 0], w) = 1 ma f ([1, 1], w) = 0 e f ([0, 0], w) = 0 [8]. I critici che hanno osservato questi difetti nei modelli lineari hanno provocato un contraccolpo contro l'apprendimento di ispirazione biologica in generale[9].

Connessionismo

Negli anni '80, la seconda ondata di ricerca sulle reti neurali è emersa in gran parte attraverso un movimento chiamato *connessionismo* o *parallel distributed processing*[10], sorto nel contesto della scienza cognitiva[11].

L'idea centrale nel connessionismo è che un gran numero di semplici unità computazionali possono ottenere un comportamento intelligente quando sono collegate

1 Amari, S. (2003). The handbook of brain theory and neural networks. MIT press.

2 Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning, MIT Press, 13-14. Traduzione: " Alcuni dei primi algoritmi di apprendimento che riconosciamo oggi erano intesi come modelli computazionali di apprendimento biologico, ovvero modelli di come l'apprendimento avviene o potrebbe avvenire nel cervello. Di conseguenza, uno dei nomi che il deep learning è passato sono le reti neurali artificiali (ANN). (...) Il termine moderno "apprendimento profondo" va oltre la prospettiva neuroscientifica dell'attuale generazione di modelli di apprendimento automatico. Fa appello a un principio più generale di apprendimento di più livelli di composizione, che può essere applicato in framework di apprendimento automatico che non sono necessariamente ispirati a livello neurale".

3 Goodfellow et al. (2016). Deep Learning.

4 McCulloch, W. S. and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 115–133.

5 Goodfellow et al. (2016). Deep Learning.

6 Ibid.

7 Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 386–408; Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan, New York.

1 Goodfellow et al. (2016). Deep Learning.

2 Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. In *1960 IRE WESCON Convention Record*, volume 4, pages 96–104. IRE, New York.

3 Goodfellow et al. (2016). Deep Learning.

4 Ibid.

5 Ibid.

6 Ibid.

7 Ibid.

8 Ibid.

9 Minsky, M. L. and Papert, S. A. (1969). *Perceptrons* . MIT Press, Cambridge.

10 Rumelhart, D. E., McClelland, J. L., and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge.

11 Goodfellow et al. (2016). Deep Learning.

in rete. Questa intuizione si applica allo stesso modo ai neuroni nei sistemi nervosi biologici e alle unità nascoste nei modelli computazionali[1].
Diversi concetti chiave sono emersi durante il movimento connessionista degli anni '80 che rimangono centrali nell'apprendimento profondo ancora oggi[2].
Uno di questi concetti è quello della rappresentazione distribuita (*distributed representation*)[3]. È l'idea che ogni input di un sistema dovrebbe essere rappresentato da molte caratteristiche, e ogni caratteristica dovrebbe essere coinvolta nella rappresentazione di molti input possibili[4].
Un altro importante risultato del movimento connessionista è stato l'uso della retro-propagazione (*back-propagation*) per addestrare reti neurali profonde con rappresentazioni interne[5]. Questo algoritmo è diventato sempre più popolare ed è attualmente l'approccio dominante all'addestramento di modelli profondi[6].

Apprendimento profondo

La terza ondata di ricerca sulle reti neurali è iniziata nel 2006 e ha reso popolare l'uso del termine apprendimento profondo per sottolineare che i ricercatori erano ora in grado di addestrare reti neurali più profonde di quanto fosse stato possibile prima e per enfatizzare l'importanza teorica del concetto di *profondità*[7].
Questa ondata di popolarità delle reti neurali continua ancora oggi, sebbene il fulcro della ricerca sul deep learning sia cambiato drasticamente nel tempo. È iniziata infatti con un focus su nuove tecniche di apprendimento non supervisionato e la capacità dei modelli profondi di generalizzare bene da piccoli set di dati, ma oggi c'è più interesse per algoritmi di apprendimento supervisionato molto più vecchi e la capacità dei modelli profondi di sfruttare grandi dataset etichettati[8].

1 Ibid.
2 Ibid.
3 Ibid.
4 Ibid.
5 Rumelhart, D., Hinton, G., and Williams, R. (1986a). Learning representations by backpropagating errors. *Nature*, 323, 533–536; LeCun, Y. (1987). *Mod`eles connexionistes de l'apprentissage*. Ph.D. thesis, Universite de Paris VI.
6 Goodfellow et al. (2016). Deep Learning.
7 Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines* . MIT Press; Delalleau, O. and Bengio, Y. (2011). Shallow vs. deep sum-product networks. In *NIPS*; Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2014a). How to construct deep recurrent neural networks. In *ICLR'2014* ; Montufar, G. (2014). Universal approximation depth and errors of narrow belief networks with discrete units. *Neural Computation*, 26.
8 Goodfellow et al. (2016). Deep Learning.

Feedforward Deep Networks

Le reti profonde feedforward, note anche come *multilayer perceptrons* (MLP), sono le reti profonde per eccellenza. Sono funzioni parametriche definite componendo insieme molte funzioni parametriche. Ciascuna di queste funzioni ha più ingressi e più uscite[1]. Nella terminologia della rete neurale, ci riferiamo a ciascuna sotto-funzione come a un livello della rete e ogni output scalare di una di queste funzioni come un'unità o talvolta come una caratteristica[2]. Anche se ogni unità implementa una mappatura o trasformazione relativamente semplice del suo input, la funzione rappresentata dall'intera rete può diventare arbitrariamente complessa[3].

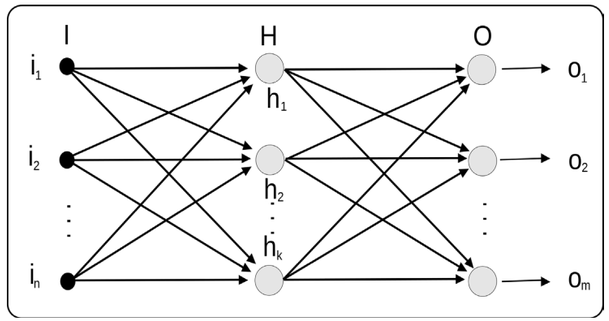


Figura A1_3 Rete neurale pienamente connessa (https://it.wikipedia.org/wiki/Rete_neurale_artificiale).

Un *multilayer perceptrons* consiste in una rete di almeno tre livelli in cui tutti i neuroni, a parte quelli di input, usano una funzione di attivazione non lineare. Se la funzione fosse lineare l'utilizzo di più livelli sarebbe inutile: combinazioni lineari di altre combinazioni lineari sono ancora combinazioni lineari.
Possiamo comunque utilizzare diversi algoritmi tradizionali di apprendimento automatico, come regressione lineare, classificatori lineari, regressione logistica e macchine kernel, che funzionano applicando una trasformazione lineare a un insieme fisso di caratteristiche[4]. Questi infatti possono anche apprendere funzioni non lineari, ma la parte non lineare è fissa. In altre parole, le funzioni non sono lineari nello spazio degli input x, ma sono lineari in qualche altro spazio predefinito[5].
Le reti neurali ci consentono proprio di apprendere nuovi tipi di non linearità. Un altro modo per vedere questa idea è che le reti neurali ci consentono di apprendere

1 Goodfellow et al. (2016). Deep Learning.
2 Ibid.
3 Ibid.
4 Ibid.
5 Ibid.

le caratteristiche fornite a un modello lineare[1].
Non tutti gli algoritmi di apprendimento profondo possono essere compresi in termini di definizione di una singola funzione deterministica come le reti profonde feedforward, ma tutti condividono la proprietà di contenere molti strati di molte unità[2]. Possiamo pensare al numero di unità in ogni livello come alla larghezza di un modello di apprendimento automatico e al numero di livelli come alla sua profondità[3]. Le reti profonde feedforward forniscono un esempio concettualmente semplice di un algoritmo che cattura i numerosi vantaggi derivanti dall'avere ampiezza e profondità significative. Le reti profonde feedforward sono anche la tecnologia chiave alla base della maggior parte delle applicazioni commerciali contemporanee dell'apprendimento profondo a grandi dataset[4].

1 Ibid.
2 Ibid.
3 Ibid.
4 Ibid.

Appendice 2*: Algoritmi genetici*

"As many other mathematical models, such as neural networks, fuzzy logic, simulated annealing techniques, particle swarm optimization, ant colony optimization and shuffled frog leaping algorithm, GA's approach was born as a tentative to explain, or at least to imitate, some phenomena related to natural systems. Nevertheless, after almost thirty years, the status of GAs in scientific research still does not seem to be well defined. In fact, in the reference literature papers on GAs in many different branches can be found – from Operation Research (OP), in which they are classified as heuristic or meta-hueristic techniques for combinatory optimization, to biology, where they are considered a suitable model to study the natural evolution"[1].

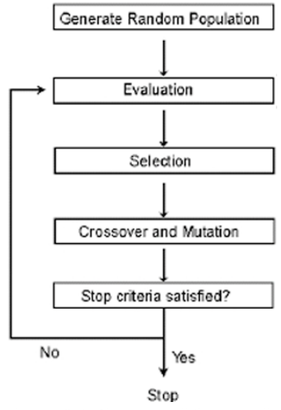


Figura A2_1 Diagramma di flusso per il processo GA. (Daru R. and Snijder, H.P.S. (1997) "GA-CAAD or AVOCAAD ? , CAAD and Genetic Algorithms for an Evolutionary Design Paradigm").

Gli algoritmi genetici, introdotti da Holland negli anni '60 e successivamente formalizzati nel 1975[2], sono stati usati come metodi di ricerca per risolvere problemi di ottimizzazione. Secondo la definizione di Goldberg, "[g]enetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics"[3]. La definizione data da Koza esplicita ulteriormente il loro funzionamento:

"The genetic algorithm is a highly parallel mathematical algorithm that transforms a set (population) of individual mathematical objects (typically fixed-length character strings patterned after chromosome strings), each with an associated fitness value, into a new population (i.e., the next generation) using operations patterned after the Darwinian principle of reproduction and survival of the fittest and after naturally occurring genetic operations (notably sexual recombination)"[4].

Pertanto, una volta creata una "popolazione" iniziale e impostato il problema, vanno ripetuti iterativamente una serie di passaggi fino a trovare il risultato voluto:
1. Valutazione della performance (fitness) di ciascun individuo;
2. Generazione di una nuova popolazione di soluzioni candidate applicando i

1 Pugnale, A. (2009). *Engineering Architecture-Advances of technological practice*, PhD Thesis, Politecnico di Torino, 25
2 Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems,* MIT Press.
3 Goldberg, D. E. (1989). *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley, 1.
4 Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection* (Vol. 1), MIT press, 18.

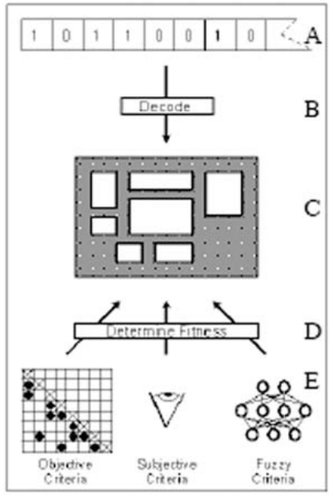


Figura A2_2 Funzione di valutazione in GA. (Daru R. and Snijder, H.P.S. (1997) "GACAAD or AVOCAAD ? , CAAD and Genetic Algorithms for an Evolutionary Design Paradigm").

seguenti tre operatori genetici (o almeno il primo): a) selezione dei migliori individui per la riproduzione per la nuova generazione (Selezione); b) ricombinazione del codice genetico degli individui e creazione di nuove soluzioni candidate (Crossover); c) applicazione di mutazioni casuali al codice genetico dei nuovi individui (Mutazione);

3. L'individuo migliore a ogni generazione (iterazione) rappresenta la soluzione, o una soluzione sub-ottimale, al problema[1].

1 Pugnale, A. (2009). *Engineering Architecture-Advances of technological practice*, PhD Thesis, Politecnico di Torino.
Sull'argomento si veda anche, oltre i già citati Goldberg e Koza, Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Bibliografia

Agraa, O. M., & Whitehead, B. (1968). "Nuisance restrictions in the planning of single-storey layouts", *Building Science*, *2*(4), 291-302.

Aish R. (2005). "Extensible computational design tools for exploratory architecture", in Kolarevic B. (ed.), *Architecture in the Digital Age: Design and Manifacturing*, Routledge.

Alexander, C (1971). "The state of the art in design methods", DMG Newsletter Vol 5 No 3, 3-7.

Alexander, C. (1964). *Notes on the Synthesis of Form*, Cambridge, Ma., Harvard University Press.

Altshuller, G. S. (1984). *Creativity as an exact science: the theory of the solution of inventive problems*, Gordon and Breach.

Amari, S. (2003). *The handbook of brain theory and neural networks*, MIT press.

Andrzejewski, D., Stork, D. G., Zhu, X., & Spronk, R. (2010, February). "Inferring compositional style in the neo-plastic paintings of Piet Mondrian by machine learning", in *Computer Vision and Image Analysis of Art* (Vol. 7531), International Society for Optics and Photonics, 75310G.

Ang, M.C., H.H.Chau, A.Mckay, A.de Pennington, "Combining evolutionary algorithms and shape grammars to generate branded product design", in: J.S. Gero (Ed.), Design Computing and Cognition '06,Springer, Netherlands, Dordrecht, 521–539.

Archer, L. B. (1979), "Whatever Became of Design Methodology?" Design Studies 1(1): 17-18.

Archer, L. B. (1965), *Systematic Method for Designers*. London, The Design Council.

Armour, G. C., & Buffa, E. S. (1963). "A heuristic algorithm and simulation approach to relative location of facilities", *Management Science*, *9*(2), 294-309.

Asimow, M. (1962), *Introduction to Design*, Englewood Cliffs, NJ., Prentice-Hall.

Arvin, S. A., & House, D. H. (2002). "Modeling architectural design objectives in physically based space planning", *Automation in Construction*, *11*(2), 213-225.

Bengio, Y., Courville, A., Vincent, P. (2013). "Representation Learning: A Review and New Perspectives", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8).

Bengio, Y., LeCun, Y., Hinton, G. (2015). "Deep Learning", *Nature*, 521 (7553).

Bengio, Y. and LeCun, Y. (2007). "Scaling learning algorithms towards AI", in L. Bottou, O. Chapelle, D. DeCoste, and J. Weston (ed.). *Large Scale Kernel Machines,* MIT Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning*, Springer.

Boden, M.A. (1991). *The Creative Mind, Myths and Mechanisms*, Wiedenfeld and Nicholson, London.

Boden, M.A. (2009). "Computer Models of Creativity"*, AI Magazine* Vol 30, No 3, 23-34.

Bois, Y. (1987)."Mondrian and the Theory of Architecture," *Assemblage 4*, 102-130.

Broadbent, G. (1973), *Design in Architecture*, Chichester, UK, John Wiley & Sons Ltd.

Broadbent, G. and A. Ward, (ed.) (1969), *Design Methods in Architecture*, London, UK, Lund Humphries.

Cache, B. (1999). "Objectile: The Pursuit of Philosophy by Other Means?", Architectural Design, vol. 69, no. 9/10, pp. 66-71.

Cache, B. (1995). *Earth Moves: The Furnishing of Territories*. Cambridge: MIT Press.

Cagan, J., Mitchell, W.J. (1993). "Optimally directed shape generation by shape annealing", Environment and Planning B: Planning and Design 20, 5–12.

Carpo, M. (ed) (2013). *The Digital Turn in Architecture 1992-2012*, John Wiley & Sons.

Carrara, G., Kalay, Y. E., & Novembri, G. (1994). Knowledge-based computational support for architectural design. *Automation in Construction*, *3*(2-3), 157-175.

Chase, S.C. (2002). "A model for user interaction in grammar-based design systems", Automation in Construction 11, 161–172.

Chan, C.-S. (1995). "A cognitive theory of style", *Environment and Planning B: Planning and Design* 22, 461.47.

Chomsky, N. (1957). *Syntactic structures*,'s-Gravenhage: Mouton. The Hague, The Netherlands.

Chouchoulas, O. (2003). *Shape Evolution.An Algorithmic Method for Conceptual Architectural Design Combining Shape Grammars and Genetic Algorithms*, Ph.D. Thesis, University of Bath.

Ciucci, G. (1996). *Giuseppe Terragni : Opera completa*, Electa.

Colakoglu, B. (2002). "An Informal Shape Grammars for Interpolation of Traditional Bosnian Hayat Houses in a Contemporary Context", Generative Art and Design Conference, Milan, Italy, 15.1-15.9.

Corbellini, G., Morassi, C. (ed). *Parametrico Nostrano*, LetteraVentidue, 109-123.

Coyne, R D ; Rosenman, M. A.; Radford, A D ; and Gero, J. S. (1987). "Innovation and Creativity in Knowledge-Based CAD", in J. S. Gero (ed.), *Expert Systems in Cornputer- Aided Design*, Amsterdam: North-Holland, 435-465.

Coyne, R. (1997). "Creativity as commonplace", *Design Studies*, *18*(2), 135-141.

Cross, N. (2007). "Forty years of design research". Design Studies, 28, pp. 1-4.

Cross N. (1993). "A History Of Design Methodology", M. J. de Vries et aL (eds.), *Design Methodology and Relationships with Science*, Kluwer Academic Publishers, 15-27.

Cross, N. (1989), *Engineering Design Methods*. Chichester, UK, John Wiley & Sons Ltd.

Dang, M., Lienhard, S., Ceylan, D., Neubert, B., Wonka, P., & Pauly, M. (2015). Interactive design of probability density functions for shape grammars. *ACM Transactions on Graphics (TOG)*, *34*(6), 1-13.

Dawson, J. W. (1961). "The Computer in Building Design", Architectural and Engineering News, vol. 3(12), 14–19.

De Bono, E. (1967). *The use of lateral thinking*. Penguin.

Delalleau, O. and Bengio, Y. (2011). Shallow vs. deep sum-product networks. In *NIPS*

Deleuze, G. (1993). *The Fold: Leibniz and the Baroque*, trans. Tom Conley, Minneapolis: University of Minnesota Press. Edizione originale francese: *Le Pli: Leibniz et le Baroque* (Paris: Minuit, 1988).

Deng, L., & Yu, D. (2014). "Deep learning: methods and applications", *Foundations and trends in signal processing*, *7*(3–4).

Duarte, J. P. (2005). "Towards the mass customization of housing: the grammar of Siza's houses at Malagueira", Environment and Planning B: Planning and Design 32(3), 347-380.

Duarte, J. P. (2001). *Customizing mass housing: a discursive grammar for Siza's Malagueira houses*, Doctoral dissertation, Massachusetts Institute of Technology.

Dylla, K., Frischer, B., Müller, P., Ulmer, A., & Haegler, S. (2008). "Rome reborn 2.0: A case study of virtual city reconstruction using procedural modeling techniques". Computer Graphics World, 16(6), 62-66.

Eastman C., Teicholz P., Sacks R., Liston K. (2008). *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*, Wiley.

Eastman, C. M., et al. (1974), "An Outline of the Building Description System", Research Report No. 50, Carnegie-Mellon Univ., Pittsburgs, Pa. Inst. Of Physical Planning.

Eastman, C. M. (1969). "Cognitive processes and ill-defined problems: A case study from design", *Proceedings of the International Joint Conference on Artificial Intelligence: IJCAI*, Vol. 69, 669-690.

Eberhard, J. P. (1962). "A Computer-Based Building Process: Its Potentials for Architecture", Architectural and Engineering News, vol. 4(12), 16–18.

Eisenman, P. (2003). Giuseppe Terragni: Transformations, Decompositions, Critiques, New York, Monacelli Press; M. Baiocchi, A. Coppola (Trad.) (2004). Giuseppe Terragni: trasformazioni, scomposizioni, critiche, Quodlibet.

Eisenman, P. (1971). "From object to relationship 2: Giuseppe Terragni. Casa Giuliani Frigerio", Perspecta: The Yale Architectural Journal number 13/14, 36-61.

Evans, B., J. Powell, R. Talbot, (ed.) (1982), *Changing Design*. Chichester, UK, John Wiley & Sons Ltd.

Felgenbaum, E.A. & Feldman, J. (eds) (1995). *Computers and thought,* MIT Press, Cambridge, MA.

Flemming, U. (1994). "Artificial Intelligence and Design: A Mid-Term Review." In G. Carrara and Y.E. Kalay (eds.). *Knowledge-Based Computer-Aided Architectural Design*, Elsevier, Netherlands, 1-24.

Flemming, U. (1994). "Case-based design in the SEED system". *Automation in construction*, *3*(2-3), 123-133.

Flemming, Ulrich (1987). "The Role of Shape Grammars in the Analysis and Creation of Designs", in: Y. Kalay (ed.), *The Computability of Design*, New York, Wiley Interscience, 245–272.

Flemming, U. (1981) "The secret of the Casa Giuliani Frigerio", Environment and Planning B, volume 8, 87-96.

Flemming, U., Coyne, R., Glavin, T., & Rychener, M. (1988). "A generative expert system for the design of building layouts – Version 2". In: Gero, J.S. (ed) *Artificial Intelligence in Engineering: Design*, Amsterdam: Elsevier.

Fox, J. (1996). "Expert systems and theories of knowledge", in M. Boden (ed.), *Artificial Intelligence*, Academic Press, 157-181.

Frew, S.R. (1980). "A survey of space allocation algorithms in use in architectural designin the past twenty years". In: Proceedings of the 17th Conference on DesignAutomation. Minneapolis, Minnesota, 165-174.

Gero, J. S. (1996). "Creativity, emergence and evolution in design". *Knowledge-Based Systems*, *9*(7), 435-448.
Gero, J.S. (1994). "Towards a model of exploration in computer-aided design", in: J.S. Gero, E. Tyugu (eds.), *Formal Design Methods for CAD*, North-Holland, Amsterdam, 315–336.

Gero, J. S. (1977). "Note on 'Synthesis and Optimization of Small Rectangular Floor Plans' of Mitchell, Steadman, and Liggett". *Environment and Planning B: Planning and Design*, *4*(1), 81-88.

Gero J.S., Kazakov, V.A. (1996). "Evolving building blocks for design using genetic engineering: a formal approach", in: Advances in Formal Design Methods for CAD, Hall, 31–50.

Gero, J.S., Louis, S.J., Kundu, S. (1994). "Evolutionary learning of novel grammars for design improvement", Artificial Intelligence for Engineering Design, Analysis and Manufacturing 8, 83–94.

Gero, J.S. and Maher, M.L. (eds.) (1993). *Modeling Creativity and Knowledge-Based Creative Design*, Lawrence Erlbaum, Hillsdale, N J.

Gero, J.S. and Maher, M.L. (1992). "Mutation and analogy to support creativity in computer-aided design", in G.N. Schmitt (ed.), CAAD Futures '91, Vieweg, Wiesbaden, 261-270.

Gips, J. (1975). *Shape Grammars and their Uses: Artificial Perception, Shape Generation and Computer Aesthetics* (Interdisciplinary Systems Research series 10). Basel and Stuttgart: Birkhäuser Verlag.

Goldberg, D. E. (1989). *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley.

Goldschmidt, G. (1991). "The dialectics of sketching". *Creativity research journal*, *4*(2), 123-143.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*, MIT Press.

Gordon, W. J. J. (1961), *Synectics*. New York, Harper & Row.

Gregory, S. A, (ed.) (1966), *The Design Method*. London, Butterworth Press.

Grobman, Y.J., Yezioro, A. & Capeluto I.G. (2009). "Computer-Based Form Generation in Architectural Design - a Critical Review", International Journal of Architectural Computing issue 04, volume 07, 535-553.

Gross, M. D. (1996). "The electronic cocktail napkin- a computational environment for working with design diagrams". *Design studies*, *17*(1), 53-69.

Hall, A D. (1962), *A Methodology for Systems Engineering*. Princeton, NJ, Van Nostrand.

Habraken, N.J. (1985). *The Appearance of the Form, four essays on the position designing takes between people and things*, Cambridge, Awater Press.

Haegler, S., Müller, P., & Van Gool, L. (2009). "Procedural modeling for digital cultural heritage". EURASIP Journal on Image and Video Processing, 2009(1), 852392.

Hansmeyer, M. and Dillenburger, B (2013). "Mesh Grammars – Procedural Articulation of Form", Open Systems: Proceedings of the 18th International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA 2013), 821-829.

Hansmeyer, M. (2010). "From Mesh to Ornament", In Future Cities: ECAADE 2010: Proceedings of the 28th Conference on Education in Computer Aided Architectural Design in Europe, September 15-18, 2010, Zurich, Switzerland, ETH Zurich, Vol. 28, vdf Hochschulverlag AG, 285-293.

Hayes-Roth, F. (1985). "Rule-based systems". *Communications of the ACM*, *28*(9), 921-932.

Hayes-Roth, F. (1984). "Knowledge-based expert systems". *Computer*, (10), 263-273.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep residual learning for image recognition". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770-778.

Heylighen,A. & Neuckermans, H. (2001). "A case base of Case-Based Design tools for architecture". Computer-Aided Design, 33(14), 1111-1122.

Hinton, G. E., & Zemel, R. S. (1994). Autoencoders, minimum description length and Helmholtz free energy. In *Advances in neural information processing systems*, 3-10.

Holland, J. H. (1998). *Emergence: from chaos to order*, Perseus, Cambridge, MA.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, MIT Press.

Jacques, R. and J. Powell, (ed.) (1981), *Design:Science:Method.* Guildford, UK, Westbury House.

Jones, J. C. (1970), *Design Methods*. Chichester, UK, John Wiley & Sons Ltd.

Jones, J. C. and D. G. Thornley, (ed.) (1963), *Conference on Design Methods*. Oxford, UK, Pergamon Press.

Kalay, Y.E. (2004) *Architecture's New Media*, Cambridge. MA: The MIT Press.

Kalay, Y. E. (1987). "Worldview: An integrated geometric-modeling/drafting system". *IEEE computer graphics and applications*, *7*(2), 36-46.

Kelly, N., & Gero, J. S. (2015). "Situated interpretation in computational creativity". *Knowledge-Based Systems*, *80*, 48-57.

Kim, S.H. (1990). *Essence of Creativity*, Oxford University Press, New York.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). "Optimization by simulated annealing", *Science*, *220*(4598), 671-680.

Knight, T.W. (2003). "Computing with emergence". *Environment and Planning B: planning and design*, *30*(1), 125-155.

Knight, T.W. (1999). "Applications in Architectural Design, and Education and Practice", Technical Report, NSF/MIT Workshop on Shape Computation.

Knight, T.W. (1999). “Shape grammars: six types,” *Environment and Planning B: Planning and Design*, 26 (1), 15-31.

Knight, T.W. (1999). “Shape grammars: five questions”. *Environment and Planning B: planning and Design*, 26(4), 477-501.

Knight, T. W. (1994). *Transformations in design: a formal approach to stylistic change and innovation in the visual arts*, Cambridge University Press.

Knight, T.W. (1992). “Designing with grammars”, in Schmitt, G.N. (ed). *Computer-Aided Architectural Design*, Wiesbaden: Verlag Viewag, 33-48.

Knight, T. W. (1989). “Transformations of De Stijl Art: The Paintings of Georges Vantongerloo and Fritz Glarner”. Environment and Planning B: Planning and Design, 16(1), 51-98.

Knight, T. W. (1981) “Languages of designs: from known to new”, Environment and Planning B 8, 213-238.

Koch, H. (1906). “Une méthode géométrique élémentaire pour l’étude de certaines questions de la théorie des courbes planes”, *Acta mathematica*, *30*, 145-174.

Kolarevic, B. (ed.) (2003). *Architecture in the digital age: design and manufacturing*, Spon Press.

Koning, H., & Eizenberg, J. (1981). “The language of the prairie: Frank Lloyd Wright's prairie houses”, Environment and planning B: planning and design, 8(3), 295-323.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection* (Vol. 1), MIT press.

Koza, J. R.; Bennett, F. H.; Andre, D.; Keane, M. A. (1996). “Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming*”, Artificial Intelligence in Design ‘96*. Springer, Dordrecht.

Langley, P., Simon, H. A., Bradshaw, G. L., & Zytkow, J. M. (1987). *Scientific discovery: Computational explorations of the creative processes*. MIT press.
LeCun, Y. (1987). *Modèles connexionistes de l’apprentissage*. Ph.D. thesis, Universite de Paris VI.

Lee, H.C., Tang, M.X. (2009). ”Evolving product form designs using parametric shape grammars integrated with genetic programming”, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23, 131–158.

Lee, J. (1998). “Modelling Mondrian’s Design Processes and Their Architectural Associations Using Multilayer Neural Networks”, in T. Sasada, S. Yamaguchi, M. Morozumi, A. Kaga, and R. Homma (eds.), *CAADRIA ‘98 [Proceedings of The Third Conference on Computer Aided Architectural Design Research in Asia]* Osaka (Japan) 22-24 April 1998, 455-464.

Lee, R. C. and Moore, J. M. (1967). “CORELAP Computerized Relationship Layout Planning.”. Journal of industrial Engineering, vol. 18(3), 195–200.

Lewes, G. H. (1875). *Problems of Life and Mind*. Osgood and Company, Boston, MA.

Lindenmayer, A. (1968). “Mathematical models for cellular interaction in development, Parts I and II”, Journal of Theoretical Biology, 18, 280–315.

Lynn, G. (1996). “Blobs (or Why Tectonics Is Square and Topology Is Groovy),” *ANY* 14, 58–62.

Lynn, G. (1998). *Folds, Bodies & Blobs: collected essays*, La lettre volée, Bruxelles.

Mandelbrot, B. (1977). Fractal Geometry of Nature, W.H. Freeman, New York.

March, L. (2011). “Forty Years of Shape and Shape Grammars, 1971 – 2011”, Nexus Network Journal Vol.13, No.1, 5-13.

March, L. (1996). “Babbage’s miraculous computation revisited”, Environment and Planning B: Planning and Design 23, 369- 376.

March, L. (1996). “Rulebound unruliness”, Environment and Planning B: Planning and Design 23, 391-399.

March, L., & Steadman, P. (1974). *The geometry of environment: an introduction to spatial organization in design*, MIT Press, 263-264.

Marinčić, Nikola (2017). *Towards Communication In Caad: Spectral Characterisation and Modelling with Conjugate Symbolic Domains,* PhD Thesis, ETH Zurich (pubblicato da Birkhauser nel 2019).

McCulloch, W. S. and Pitts, W. (1943). “A logical calculus of ideas immanent in nervous activity”. *Bulletin of Mathematical Biophysics*, 5.

Michalek, J, Choudhary, R. Papalambros, P. (2002). “Architectural layout design optimization”. Engineering Optimization 34(5), 461-484.

Migayrou F. (ed) (2004), *Architectures non standard*, Centre Pompidou, Parigi.

Minsky, M. L. and Papert, S. A. (1969). *Perceptrons,* MIT Press, Cambridge.

Mill, J. S. (1843). *System of Logic*, Longmans, Green, Reader, and Dyer, London.

Mitchell, W. J., Steadman, J. P., & Liggett, R. S. (1976). “Synthesis and optimization of small rectangular floor plans”. *Environment and Planning B: Planning and Design*, *3*(1), 37-70.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Mitchell, T. (1997). *Machine Learning*. New York: McGraw Hill.

Mondal, J. (2019). “Procedural Generation of Mondrian’s Neo-plastic 2D Compositions for Stylistic Massing Extrusion using Shape Grammar”, Proceedings of International Conference on Architecture Pedagogy (ICAP-2019) 11 - 13 November 2019, Faculty of Architecture & Ekistics, Jamia Millia Islamia, New Delhi, India.

Montufar, G. (2014). “Universal approximation depth and errors of narrow belief networks with discrete units”, *Neural Computation*, 26.

Moore, G. T., (ed.) (1970), *Emerging Methods in Environmental Design and Planning*, Cambridge, Ma., MIT Press.

Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J. (2017). “3d bounding box estimation using deep learning and geometry”, IEEE Conference on Computer Vision and Pattern Recognition.

Müller, P. & Parish, Y. (2001). “Procedural modelling of cities”. Proceedings of the 28th annual con-

ference on Computer Graphics and Interactive Techniques – SIGGRAPH '01. New York, USA. New York: ACM Press; 301–308.

Müller, P., Vereenooghe, T., Wonka, P., Paap, I., & Van Gool, L. (2006).“Procedural 3D Reconstruction of Puuc Buildings in Xkipché”, The 7th International Symposium on Virtual Reality, Archeology and Cultura Heritage VAST 2006, Eurographics Symposium Proceedings, Aire-la-Ville, 139-146.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). "Procedural modeling of buildings". In ACM SIGGRAPH 2006 Papers, 614-623.

Negroponte, N. (1975), *Soft Architecture Machines*, The MIT Press, Cambridge.

Negroponte, N. (1970). *The Architecture Machine*, The MIT Press, Cambridge.

O'Neill, M., J.McDermott, J.M.Swafford, J.Byrne, E.Hemberg, A.Brabazon, E. Shotton, C.McNally, M.Hemberg (2010). “Evolutionary design using grammatical evolution and shape grammars: designing a shelter”, International Journal of Design Engineering 3.

Oosterhuis, K. (2002). *Programmable Architecture*, L'Arcaedizioni.

Oosterhuis, K. (1998). 'Salt Water Live, Behaviour of the Salt Water Pavilion', Stephen Perrella (guest-editor), *Hypersurface Architecture*, *AD* Profile 133, *AD* 68.

Osborn, A F. (1963), *Applied Imagination - Principles and Procedures of Creative Thinking*. New York, Scribener's Sons.

Oxman, R. E. (2002). “The thinking eye: visual re-cognition in design emergence”, *Design Studies*, *23*(2), 135-164.

Oxman, R. E. (1994). “Precedents in design: a computational model for the organization of precedent knowledge”. *Design studies*, *15*(2), 141-157.

Oxman, R. E. (1992). “Multiple Operative And Interactive Modes In Knowledge-Based Design Systems”. In: Kalay, Y.E. (ed) *Principles of Computer-Aided Design: Evaluating and Predicting Design Performance*, John Wiley&Sons, 125-143.

Papamichael, K., LaPorta, J., & Chauvet, H. (1997). Building Design Advisor: automated integration of multiple simulation tools. *Automation in construction*, *6*(4), 341-352.

Park, H-J. (2020). “Stylistic reproductions of Mondrian's composition with red, yellow, and blue”, in D. Holzer, W. Nakapan, A. Globa, I. Koh (eds.), *RE: Anthropocene, Design in the Age of Humans - Proceedings of the 25th CAADRIA Conference - Volume 2, Chulalongkorn University, Bangkok, Thailand, 5-6 August 2020,* 133-142.

Partridge, D. (1996). “Representation of Knowledge”, in M. Boden (ed.), *Artificial Intelligence*, Academic Press, 55- 87.

Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2014). “How to construct deep recurrent neural networks”, in *ICLR'2014*.

Pohl, J., & Myers, L. (1994). “A distributed cooperative model for architectural design”. *Automation in Construction*, *3*(2-3), 177-185.

Prusinkiewicz, P. & Lindenmayer, A. (1991). *The Algorithmic Beauty of Plants*, Springer Verlag.

Prusinkiewicz, P., James, M., & Měch, R. (1994, July). “Synthetic topiary”. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 351-358.

Pugnale, A. (2012). “Engineering Architecture: Come il virtuale si fa reale”, *Bloom*, n.14.

Pugnale, A. (2009). *Engineering Architecture-Advances of technological practice*, PhD Thesis, Politecnico di Torino.

Reddy, G. & Cagan, J. (1995). “An Improved Shape Annealing Algorithm For Truss Topology Generation.” ASME. *J. Mech. Des.* June 1995; 117(2A), 315–321.

Riesbeck, C. K., & Schank, R. C. (1989). *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ, US.

Ritchie, D., Thomas, A., Hanrahan, P., & Goodman, N. (2016). “Neurally-guided procedural models: Amortized inference for procedural graphics programs using neural networks”. In *Advances in neural information processing systems*, 622-630.

Ritchie, D., Mildenhall, B., Goodman, N. D., & Hanrahan, P. (2015). “Controlling procedural modeling programs with stochastically-ordered sequential monte carlo”, *ACM Transactions on Graphics (TOG)*, *34*(4), 1-11.

Ritchie, D., Lin, S., Goodman, N. D., & Hanrahan, P. (2015, May). “Generating design suggestions under tight constraints with gradient-based probabilistic programming”. In *Computer Graphics Forum*, Vol. 34, No. 2, 515-526.

Rittel, H. (1973), “The State of the Art in Design Methods.” Design Research and Methods (Design Methods and Theories) 7(2): 143-147.

Rittel, H. and M. Webber (1973), “Dilemmas in a General Theory of Planning.” Policy Sciences 4: 155-169.

Rocker, I.M. (2008). “Architectures of the Digital Realm: Experimentations by Peter Eisenman | Frank O. Gehry”, *Die Realität des Imaginären. Architektur und das digitale Bild*, 10 Internationales Bauhaus Kolloquium, Weimar, Bauhaus- Universität Weimar, 249-262.

Rosenblatt, F. (1958). “The perceptron: A probabilistic model for information storage and organization in the brain”, *Psychological Review*, 65.

Rosenblatt, F. (1962). *Principles of Neurodynamics*, Spartan, New York.

Roozenburg, N. and N. Cross (1991). “Models of the Design Process: integrating across the disciplines.” Design Studies 12(4): 215-220.

Rowe, P.G. (1987). *Design Thinking*, MIT Press, Cambridge, MA.

Ruiz-Montiel M., Boned J., Gavilanes J., Jiménez E., Mandow L., Pérez-de-la-Cruz J.L. (2013) “Design with shape grammars and reinforcement learning”, Advanced Engineering Informatics 27, 230–245.

Rumelhart, D. E., McClelland, J. L., and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, Cambridge.

Rumelhart, D., Hinton, G., and Williams, R. (1986). “Learning representations by backpropagating errors”. *Nature*, 323, 533–536.

Russell, S. (1996). “Machine Learning”, in Boden, M. (ed.). *Artificial Intelligence,* Academic Press.

Salomon, D. (2011). *The Computer Graphics Manual*, Texts in Computer Science, David Gries and Fred. B. Schneider (ed.), vol. 2, London: Springer Science & Business Media.

Sass, L. (2005). “Wood frame Grammar: CAD scripting a wood frame house, CAAD Futures Conference, Vienna, 1-10.

Schmidhuber, J. (2015). “Deep Learning in Neural Networks: An Overview”, *Neural Networks*, 61.

Schnier, T., & Gero, J. S. (1998). “From Frank Lloyd Wright to Mondrian: Transforming Evolving Representations”, *Adaptive Computing in Design and Manufacture*, Springer, London, 207-219.

Schnier, T. & Gero, J. S. (1996). “Learning genetic representations as alternative to hand-coded shape grammars”, *in* J. S. Gero & F. Sudweeks (eds), *Artificial Intelligence in Design ‘96*, Kluwer, Dordrecht, 39-57.

Schon, D. A. (1983). *The Reflective Practitioner: How Professionals Think in Action*. Basic Books: N. Y.

Schwarz, A., Berry, D. M., & Shaviv, E. (1994). Representing and solving the automated building design problem. *Computer-aided design*, *26*(9), 689-698.

Sdegno A (2016). “Computer Aided Architecture: origini e sviluppo”, DISEGNARECON volume 9, n.16, 4.1-4.6: 4.2.

Sdegno, A. (2013) “Sketchpad: sulla nascita del disegno digitale”, Disegnare Idee Immagini, 46, XXIII, 74-81.

Sdegno, A. (2009). “Breve storia della rappresentazione numerica”, in R. Migliari, *Geometria descrittiva, Volume I – Metodi e costruzioni*, CittàStudi-De Agostini, Novara, 245-252.

Shaviv, E., & Gali, D. (1974). “A model for space allocation in complex buildings: an aproach”. *Building Industrial 6*, 493-518.

Shea, K., Cagan, J. (1997). “Innovative dome design: applying geodesic patterns with shape annealing”, Artificial intelligence for engineering design, analysis and manufacturing 11, 379–394.

Shea, K., Cagan, J. (1999). “Languages and semantics of grammatical discrete structures”, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 13, 241–251.

Shea, K., Cagan, J. (1999). “The design of novel roof trusses with shape annealing: assessing the ability of a computational method in aiding structural designers with varying design intent”, Design Studies 20, 3–23.

Shelden, D. (2002). “Digital Surface Representation and the Constructibility of Gehry's Architecture, PhD Disseration, MIT.

Simon, H.A. (1973). “The structure of ill structured problems”, Artificial Intelligence 4, 181–201.

Sounder, J. J. And Clark, W. E. (1963). “Computer Technology: A New Tool for Planning”. *AIA Journal*, October: 97–106.

Souder, J. J., Clark, W. E., Elkind, J. I., Brown, M. B. (1964). *Planning for Hospitals. A System Approach Using Computer- Aided Techniques*, American Hospital Association, Chicago.

Spillers, W. R, (ed.) (1974), *Basic Questions of Design Theory*. Amsterdam/New York, North-Holland/Elsevier.

Spuybroek, L. (ed.) (2009). *Research & Design: The Architecture of Variation*, Thames & Hudson.

Spuybroek, L. (1998). ‘Motor Geometry’, *Hypersurface Architecture*, Stephen Perrella (guest-editor), *AD* Profile 133, *AD* 68, 49–55.

Stava, O., Pirk, S., Kratt, J., Chen, B., Měch, R., Deussen, O., & Benes, B. (2014, September). Inverse procedural modelling of trees. In *Computer Graphics Forum*, Vol. 33, No. 6, 118-131.

Sternberg, R. (ed.) (1988) *The Nature of Creativity*, Cambridge University Press, Cambridge.

Stiny, G. (1975). *Pictorial and Formal Aspects of Shape and Shape Grammars: On Computer Generation of Aesthetic Objects* (Interdisciplinary Systems Research series 13). Basel and Stuttgart: Birkhäuser.

Stiny, G. (1976). “Two exercises in formal composition”. *Environment and Planning B: Planning and Design*, *3*(2), 187-210.

Stiny, G. (1980). “Introduction to shape and shape grammars”, *Environment and Planning B: Planning and Design,* 7, 343-351.

Stiny, G. (1980). “Kindergarten grammars: designing with Froebel’s building gifts,” *Environment and Planning B: Planning and Design,* 3, 409-462.

Stiny, G. (1994). “Shape rules: closure, continuity, and emergence”, *Environment and Planning B*, 21, S49-S78.

Stiny, G. (2006). *Shape: talking about seeing and doing*. MIt Press.

Stiny, G. and Gips, J. (1972). “Shape Grammars and the Generative Specification of Painting and Sculpture,” in C. V. Freiman (ed.). *Information Processing 71*, North Holland, Amsterdam, 1460-1465.

Stiny, G. and J. Gips (1978). *Algorithmic Aesthetics*, University of California Press, Berkeley, CA.

Stiny, G. and L. March (1981). “Design machines”. *Environment and Planning B: Planning and Design*, 8, 245-256.

Stiny, G., & Mitchell, W. J. (1980). “The grammar of paradise: on the generation of Mughul gardens”. *nvironment and Planning B: Planning and Design*, 7(2), 209-226.

Stiny, G, & Mitchell, W. J. (1978). “The Palladian grammar”. *Environment and Planning B: Planning and Design*, 5, 5-18.

Studer R., Benjamins V.R., Fensel D. (1998). “Knowledge engineering: principles and methods”, Data and Knowledge Engineering 25, 161–197.

Sutherland, I. E. (1963), *Sketchpad. A Man-Machine Graphical Communication System*, Ph.D. Thesis, Massachusetts Institute of Tecnology, Cambridge, Mass.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.

Talton, J. O., Lou, Y., Lesser, S., Duke, J., Měch, R., & Koltun, V. (2011). Metropolis procedural model-

ing. *ACM Transactions on Graphics (TOG)*, *30*(2), 1-14

Tepavčević, B., Stojaković, V. (2012). "Shape Grammar in contemporary architectural theory and design", *Architecture and Civil Engineering* Vol. 10, No 2, 169-178.

Teresko, J. (1993). "Parametric Technology Corp.: Changing the way Products are Designed." *Industry Week*

Weisberg, R.W. (1986). *Creativity: Genius and Other Myths*, W.H. Freeman, New York.

Whitehead, B. and Eldars, M. Z. (1964). "An Approach to the Optimum Layout of Single-Story Buildings.". Architects' Journal, 17, 1373–1379.

Wiggins, G. A. (2006). Searching for computational creativity. *New Generation Computing*, *24*(3), 209-222.

Widrow, B. and Hoff, M. E. (1960). "Adaptive switching circuits". In *1960 IRE WESCON Convention Record*, volume 4, IRE, New York.

Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. (2003). "Instant architecture", ACM Transactions on Graphics 22, 3, 669–677.

Zeng, H., Wu, J., & Furukawa, Y. (2018). "Neural procedural reconstruction for residential buildings". In *Proceedings of the European Conference on Computer Vision (ECCV)*, 759-775.