Le direzioni della ricerca logica in Italia: Logica e Informatica*

Felice Cardone
Dipartimento di Informatica
Università di Torino

felice.cardone@unito.it

1 Introduzione

Che ci sia una relazione stretta tra logica ed informatica si può capire anche solo dal numero di pubblicazioni i cui titoli accostano le due discipline con preposizioni che suggeriscono vari gradi di vicinanza, di subordinazione o di cooperazione.[1] Sfogliandole, oltre all'utilizzo di nozioni della teoria della calcolabilità nella elaborazione di modelli di calcolo, troviamo risultati che vanno dalla deduzione automatica alle logiche non-classiche per applicazioni di intelligenza artificiale, dai sistemi formali per la dimostrazione di correttezza di programmi agli ambienti software che assistono nella costruzione di dimostrazioni, e ancora dalle applicazioni della teoria dei modelli finiti in teoria della complessità alla logica costruttiva.

È fuori discussione, in una sintesi sulle direzioni della ricerca nell'area al confine tra logica e informatica, dedicare anche solo qualche cenno a ciascuno di questi sviluppi.[2] Preferiamo invece abbandonare dall'inizio qualsiasi ambizione di completezza e concentrare la nostra attenzione su un ambito in cui la logica entra in contatto con le problematiche sollevate dalle recenti

*Apparso in: Le direzioni della ricerca logica in Italia a cura di Hykel Hosni, Gabriele Lolli e Carlo Toffalori, Edizioni della Normale, Pisa, 2015, pagg. 85–115, ISBN: 978-88-7642-570-7.

[1] Così abbiamo, per esempio, "Logic in Computer Science", "Logic and Computer Science", "Logic for Computer Science", "Logic from Computer Science",...

[2] La limitazione geografica non è di grande aiuto nel ridurre la dimensione del materiale da analizzare: in molti settori di quest'area di ricerca il contributo degli scienziati italiani (operanti in istituzioni italiane o straniere) è stato determinante per il loro sviluppo a livello internazionale.

trasformazioni nell'utilizzo del calcolo, che hanno portato ad una concezione interattiva in cui la funzione del calcolatore non è più (solo) quella di elaborare dati per produrre risultati, ma (soprattutto) quella di coordinare agenti umani o artificiali.

Ci muoveremo quindi nell'ambito più ristretto dei fondamenti dell'informatica, intesa come disciplina il cui oggetto è lo studio generale dei processi computazionali di trasformazione dell'informazione. Qui, un primo importante contributo della logica consiste nell'avere fornito un bagaglio di nozioni e di tecniche che pervade l'intera disciplina a qualsiasi livello. Per esempio, la necessità di descrivere insiemi generati mediante regole meccaniche ha portato a formulare varie versioni di una nozione generale di *sistema formale* che ha avuto un ruolo centrale prima nell'elaborazione delle grammatiche formali (Chomsky, 1959), poi nell'isolare forme di induzione adatte allo studio delle strutture utilizzate in programmazione, tipicamente l'induzione strutturale (Burstall, 1969). La stessa nozione generale è anche alla base della formulazione di sistemi per l'inferenza di tipi (Curry e Feys, 1958), e della costruzione di formalismi per la semantica operazionale dei linguaggi di programmazione ispirati dalle regole della deduzione naturale di Gentzen (Plotkin, 2004; Burstall e Honsell, 1988). Per contrasto, essa facilita la caratterizzazione di quelle nozioni che invece non ammettono descrizione induttiva, come le varie forme di circolarità la cui espressività nell'ambito del paradigma funzionale di programmazione à stata riconosciuta fin da una fase precoce (Kahn, 1974).

L'idea di sistema formale e di generazione induttiva si intersecano in più occasioni con lo sviluppo di una nozione altrettanto generale di *riduzione* attraverso la quale esprimere la dinamica di sistemi di calcolo, studiandone le proprietà indipendenti dalla natura degli oggetti (termini, stati, o configurazioni di vario genere) ai quali si applica. Nata nell'ambito dei primi studi della riduzione nel λ-calcolo (Church e Rosser, 1936) e subito riconosciuta come oggetto di studio indipendente attraverso tecniche combinatorie utilizzate nella topologia algebrica dell'epoca (Newman, 1942), questa relazione e le sue proprietà astratte mantengono la loro rilevanza attraverso le applicazioni nella descrizione operazionale dei linguaggi di programmazione sequenziali. Una nozione di riduzione interviene anche nella descrizione dell'evoluzione di sistemi distribuiti permettendo, attraverso una rinnovata considerazione della sua natura topologica, una elegante formalizzazione del fenomeno della concorrenza.

Dal punto di vista semantico, che è l'ambito entro il quale si muoverà la nostra esposizione, l'influenza sulla teoria della programmazione del bagaglio concettuale sviluppato dalla ricerca sui linguaggi formali della logica ha una

manifestazione fondamentale nell'impiego di tecniche composizionali nelle definizioni semantiche, come accade in teoria dei modelli da Tarski in poi.

Proprio un allievo di Tarski, Dana Scott, scopre nel 1969 una tecnica generale per la costruzione di modelli del λ-calcolo. Questo è l'atto iniziale della semantica denotazionale dei linguaggi di programmazione e, più ancora, di una teoria astratta del calcolo che si basa sull'intuizione di strutture ordinate di elementi parziali (i *domini di Scott*, appunto), in cui la relazione d'ordine corrisponde ad una nozione puramente qualitativa di informazione presente negli elementi. Lo sviluppo dell'assiomatizzazione è ben noto (Amadio e Curien, 1998), e proprio nel nostro Paese ha dato origine ad una vasta attività di ricerca che ha contribuito in molti modi allo sviluppo della disciplina. Da un lato abbiamo un'indagine sistematica delle connessioni dei linguaggi di programmazione e dei sistemi di tipi ad essi associati con gli insiemi ordinati, la topologia, la calcolabilità in una serie di importanti contributi di Longo, Giannini, Martini, Moggi e Rosolini (Giannini e Longo, 1984; Longo e Martini, 1986; Moggi, 1988; Robinson e Rosolini, 1988; Longo, 2004; Longo e Moggi, 1991). È in questo contesto che prende forma una teoria sintetica dei domini in cui un ruolo tecnico centrale è giocato dalla nozione di *dominanza*, introdotta da Rosolini (Rosolini, 1986) per ottenere una versione astratta della nozione di dominio di una funzione ricorsiva parziale. Dagli stessi sviluppi tecnici ha origine il progetto di unificazione della semantica denotazionale basato sulla nozione categoriale di monade in (Moggi, 1991), implementata ed usata sistematicamente nel linguaggio Haskell, in particolare nella strutturazione delle primitive di input/output. Parallelamente viene sviluppata la rappresentazione dei domini di Scott (Coppo *e altri*, 1983, 1984) sulla base di teorie dei tipi per il λ-calcolo nate indipendentemente (Coppo *e altri*, 1981) per la caratterizzazione di proprietà operazionali dei λ-termini. Questi ultimi sviluppi hanno una relazione stretta con forme di *pointless topology* motivate dall'osservazione che la nozione di punto di uno spazio topologico non è costruttiva. Nel contesto della teoria dei tipi di Martin-Löf (1984) questa idea ha dato luogo ad una nozione di spazio formale intuizionistico (Sambin, 1987) studiato intensivamente dalla scuola di Padova (Sambin, 2003).[3]

Esistono alcuni assi che possono fornire coordinate di riferimento per organizzare il nostro discorso. Si è già parlato di una concezione interattiva del calcolo: prendendo le mosse dall'idea di *interazione* si cercherà di aggregare intorno ad essa alcuni sviluppi recenti della ricerca in informatica teorica in cui è prominente il contributo italiano. Emergerà in modo naturale un

[3]Si veda il contributo di Giovanni Sambin al presente volume.

altro tema, quello della *geometria*, che sta assumendo una sempre maggiore rilevanza come strumento tecnico e concettuale di unificazione di direzioni di ricerca finora indipendenti, ad esempio la vasta area rappresentata dalla teoria dei sistemi concorrenti (Pratt, 1991), i recenti modelli per l'uguaglianza intensionale nella teoria dei tipi di Martin-Löf basati su classi di gruppoidi utilizzati nella teoria delle omotopie di ordine superiore (Awodey, 2012) e la teoria delle n-categorie (Leinster, 2004) interessanti anche per la fisica matematica (Baez e Lauda, 2011). Queste direzioni convergono verso l'impiego strutture di ordine superiore nel calcolo, tendenza che si osserva anche in aree della logica abbastanza distanti, per esempio la formulazione di una teoria della dimostrazione per le modalità (Masini, 1992) ed il lavoro di Alessio Guglielmi e dei suoi collaboratori sulla *deep inference* (Guglielmi, 2007; Guiraud, 2006).

2 Interazione

Il tema dell'*interazione* si offre in modo naturale come candidato per una visione unificata di alcuni percorsi notevoli all'interno del panorama attuale della ricerca: dallo studio dei modelli della concorrenza (Milner, 1993) a quelli dei sistemi di logica lineare (Girard, 1989) emerge la centralità di nozioni che sono tutte riconducibili a forme generali di interazione tra entità computazionali. Si potrebbe prendere l'emergere di questa nozione di interazione come segno distintivo della formazione di una nuova nozione di calcolo, o di fenomeno computazionale, che non è estranea ad influenze che provengono dai recenti sviluppi della biologia dei sistemi (Cardelli, 2008) e della fisica quantistica (Coecke e Duncan, 2011). Vale la pena soffermarsi su alcuni aspetti dell'evoluzione di questo concetto ampio di interazione: la prospettiva storica ci consentirà di giustificare meglio la scelta di questa nozione come uno dei *leitmotiv* della ricerca al confine tra logica ed informatica teorica.

2.1 Verso un'ecologia del calcolo

Il cammino verso una nozione generale di interazione inizia dallo studio dei sistemi aperti e dalla simmetria tra sistema ed ambiente che assume prominenza in questo ambito. Dal punto di vista epistemologico, è per i sistemi aperti che diventa rilevante la distinzione tra metodo di studio funzionale e metodo comportamentistico, enunciata per la prima volta nel classico (Rosenblueth *e altri*, 1943, p. 18):

> Given any object, relatively abstracted from its surroundings for study, the behavioristic approach consists in the examination of the output of the object and of the relations of this output to the input. By output is meant any change produced in the surroundings by the object. By input, conversely, is meant any event external to the object that modifies this object in any manner [...] In a functional analysis, as opposed to a behavioristic approach, the main goal is the intrinsic organization of the entity studied, its structure and its properties; the relations between the object and the surroundings are relatively incidental.

Nel contesto più particolare della teoria dei sistemi di calcolo, il metodo comportamentistico trova naturale espressione nell'idea di *esperimento* (*Gedankenexperiment*) proposto in (Moore, 1956) per le macchine sequenziali:

> This paper is concerned with finite automata from the experimental point of view [...] it is concerned with what kinds of conclusions about the internal conditions of a finite machine it is possible to draw from external experiments [...] A copy of the sequential machine being observed experimentally will receive successively input symbols from the experimenter [...] This sequence of input symbols, together with the sequence of output symbols, will be called the outcome of the experiment (ibid., pp. 129–130)

Vediamo alcune applicazioni esemplari di questa idea in contesti computazionali.

2.1.1 Equivalenza di Myhill-Nerode

Nel teorema di Myhill-Nerode (Myhill, 1957; Nerode, 1958) gli esperimenti sono usati per caratterizzare una relazione di equivalenza su sequenze di simboli di ingresso: due parole sono equivalenti (relativamente ad un linguaggio L) quando sono indiscernibili, cioè quando non possono essere separate da alcun esperimento. Più precisamente, dato un linguaggio L e due parole u, v

$$u \backsim_L v =_{\text{def}} \forall\ w\ (uw \in L \iff vw \in L).$$

Al di là dell'utilizzo tecnico di questa equivalenza nella teoria degli automi per caratterizzare i linguaggi regolari e gli automi minimali ad essi associati, la sua definizione ha fornito un modello per altre relazioni di indiscernibilità basate su specifiche interpretazioni del repertorio di esperimenti disponibili all'osservatore di un sistema.

2.1.2 Equivalenza osservazionale

Nel suo lavoro sulla semantica della concorrenza, Robin Milner ha insistito su due idee portanti del suo stile di modellizzazione: la nozione di *osservazione* e l'idea strettamente associata di *estensionalità* (Milner, 1980, p. 2):

> we aim to describe a concurrent system fully enough to determine exactly what behaviour will be seen or experienced by an external observer. Thus the approach is thoroughly *extensional*; two systems are indistinguishable if we cannot tell them apart without pulling them apart.

Nel contesto generale dei linguaggi di programmazione, la nozione di *equivalenza osservazionale* ha origine diretta nel λ-calcolo, dove viene introdotta nel 1968 da Morris (1968, p. 50), che usa il teorema che Böhm aveva appena dimostrato (Böhm, 1968) per caratterizzare l'equivalenza osservazionale come la massima tra le congruenze sui λ-termini che estendono la $\beta\eta$-convertibilità.

Milner riprende questa definizione inserendola nella concezione denotazionale della semantica dei linguaggi di programmazione. Una semantica deve fornire un modo di ottenere i significati dei costrutti di un linguaggio B per composizione dei significati dei componenti (Milner, 1975, p. 167). Questo equivale alla richiesta che la funzione semantica $\mathcal{M}$ sia un omomorfismo tra struttura sintattica e modello; l'identità di interpretazione relativamente a $\mathcal{M}$ determina allora una congruenza sui programmi. Se $\mathcal{E}$ è una semantica operazionale di B, si può analogamente considerare una nozione operazionale di equivalenza tra programmi, $e \backsim_{\mathrm{op}} e'$ se e solo se $\mathcal{E}(e) = \mathcal{E}(e')$, la quale in genere non è una congruenza perché la semantica operazionale in genere non è composizionale. La massima congruenza che approssima $\sim$ è ottenuta allora imponendo la chiusura per contesti: definendo una *osservazione* di un programma e come il risultato (se esiste) della valutazione di e in un contesto $C[\ \ ]$, la relativa equivalenza *osservazionale* è

$$e \approx_{\mathrm{op}} e' =_{\mathrm{def}} \forall C[\ \ ]\ (C[e] \backsim_{\mathrm{op}} C[e']).$$

In generale, si dice che $\mathcal{M}$ è *fully abstract* se

$$\mathcal{M}(e) = \mathcal{M}(e') \Leftrightarrow e \approx_{\mathrm{op}} e'.$$

Il problema di trovare una semantica denotazionale fully abstract per il linguaggio paradigmatico PCF[4] ha dominato la ricerca in semantica per i due

[4]Si tratta di un λ-calcolo con tipi semplici, primitive aritmetiche ed un operatore di punto fisso per funzioni di tipo $\sigma \to \sigma$, per ogni tipo σ.

decenni successivi la sua formulazione in (Milner, 1977; Plotkin, 1977). L'equivalenza osservazionale è ovviamente estensionale, essendo basata esclusivamente sul comportamento osservabile dei programmi, esattamente come accade nell'analisi dei sistemi concepiti come scatole nere che trova espressione nell'assetto sperimentale di Moore.

Un altro contributo in questo ambito proviene ancora da Milner (1980), che pone alla base del suo *Calculus of Communicating Systems (CCS)* la simmetria che risulta dal rappresentare sistema ed osservatore nello stesso linguaggio (ibid., pag. 19):

> This suggests an obvious *symmetry*; we would like to represent the observer as a machine, then to represent the composite observer/machine as a machine, then to understand how *this* machine behaves for a new observer.

È naturale tentare di utilizzare il punto di vista osservazionale per caratterizzare formalmente l'equivalenza di processi concorrenti, codificati come termini di un calcolo che ha come astrazioni fondamentali *azioni* atomiche e operatori di un'algebra che permettono di costruire rappresentazioni di processi complessi. In particolare, assumiamo che siano presenti almeno un processo inerte $\mathbf{0}$, un processo della forma $a.P$ per ogni processo P ed ogni azione a in un insieme Act di azioni atomiche, per ogni coppia di processi P, Q un processo $P + Q$ che esprime una scelta non deterministica tra i due ed un processo $P \mid Q$ che esprime la composizione parallela di P e Q. Il comportamento di un processo è descritto regole di transizione etichettate da azioni, per esempio:

$$\frac{}{a.P \xrightarrow{a} P} \qquad \frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'} \qquad \frac{Q \xrightarrow{a} Q'}{P + Q \xrightarrow{a} Q'}$$

$$\frac{P \xrightarrow{a} P'}{P \mid Q \xrightarrow{a} P' \mid Q} \qquad \frac{Q \xrightarrow{a} Q'}{P \mid Q \xrightarrow{a} P \mid Q'}$$

Esistono due ragioni per la difficoltà di caratterizzare l'equivalenza di processi. In primo luogo, la scelta della nozione appropriata di osservazione dipende strettamente dall'obiettivo dello studio dei comportamenti di un sistema. Così esistono molte nozioni estensionali di equivalenza tra processi, associate a scelte di specifici assetti sperimentali. Risultati pionieristici in quest'area di indagine sono dovuti a De Nicola (De Nicola e Hennessy,

1984).[5] Per fare un esempio elementare, il processo $a.(b.\mathbf{0} + c.\mathbf{0})$ che esegue un'azione a e poi esegue un'azione b oppure esegue un'azione c dà luogo alle stesse sequenze di azioni del processo $a.b.\mathbf{0} + a.c.\mathbf{0}$: i due processi hanno le stesse *tracce* $\{ab, ac\}$, ed in questo senso sono equivalenti. Tuttavia, nello stato iniziale in cui l'azione a deve ancora essere eseguita, nel primo caso ogni azione eseguibile porta ad uno stato in cui si può scegliere tra eseguire b o eseguire c. Al contrario, nel secondo processo, l'esecuzione di una delle due copie dell'azione a porta ad uno stato in cui o non è più possibile eseguire b, o non è più possibile eseguire c: i due processi non sono quindi *bisimili*, dove la bisimulazione (Park, 1981) è la nozione canonica di equivalenza osservazionale tra processi.[6] Formalmente, se $\mathcal{P}$ è l'insieme dei processi, una *bisimulazione (forte)* è una relazione binaria $R \subseteq \mathcal{P} \times \mathcal{P}$ per cui si abbia, ogni volta che $P\ R\ Q$:

$$P \xrightarrow{a} P' \Rightarrow \exists Q'(Q \xrightarrow{a} Q' \wedge P'\ R\ Q');$$
$$Q \xrightarrow{a} Q' \Rightarrow \exists P'(P \xrightarrow{a} P' \wedge P'\ R\ Q').$$

Due processi sono *bisimili* se tra essi esiste una bisimulazione.

2.1.3 Oggetti infiniti

La seconda delle difficoltà alle quali abbiamo fatto riferimento deriva dalla possibilità di definire processi che comportano l'esecuzione di infinite azioni: questo richiede l'esplorazione di un nuovo ambito di problematiche legate alla caratterizzazione dell'uguaglianza tra oggetti infiniti.

Il metodo di dimostrazione conosciuto come *induzione strutturale* (Curry e Feys, 1958) si applica ad insiemi (per i quali si usa la notazione $I(B, \Sigma)$) caratterizzati induttivamente come i più piccoli (rispetto all'inclusione) che contengono un insieme B di *elementi di base* e sono chiusi rispetto alle *operazioni* di una segnatura Σ. Questa tecnica di induzione è fondamento della trattazione dei linguaggi simbolici in logica, e a partire da (Burstall, 1969) è stata usata sistematicamente nella dimostrazione di correttezza di programmi (funzionali) definiti su strutture di dati costruite induttivamente, come le liste e gli alberi finiti. L'essenza del principio di induzione strutturale consiste nel fatto che ad ogni elemento di un insieme della forma $I(B, \Sigma)$ è

[5] Si rinvia a (De Nicola, 2011a,b) per una breve rassegna del lavoro di ricerca sulle equivalenze comportamentali nell'ambito delle algebre di processi.

[6] Davide Sangiorgi ha studiato estesamente la nozione di bisimulazione in (Sangiorgi, 2012). Si raccomanda anche la lettura dello studio di Sangiorgi sulle origini di questa nozione (Sangiorgi, 2009).

associato un albero *ben fondato*, cioè privo di rami infiniti, che rappresenta la sua costruzione a partire da elementi di base a cui sono associate le foglie.

Questa caratterizzazione del principio di induzione strutturale permette di mettere in evidenza, per contrasto, quale è invece la caratteristica di termini come per esempio il termine P di CCS definito dall'equazione $P = a.P$ che rappresenta un processo che esegue una sequenza infinita di azioni a: l'albero di costruzione di P ha la forma

$$\dfrac{a \quad \dfrac{a \quad \dfrac{a \quad \vdots}{P}}{P}}{P}$$

con un evidente ramo infinito. Questo fenomeno si riproduce anche nel caso di strutture di dati infinite come gli stream $\texttt{ones} = \texttt{1} : \texttt{ones}$ o lo stream dei numeri naturali, che soddisfa l'equazione $\texttt{nats} = \texttt{map}\ (+\texttt{1})\ \texttt{ones}$, oppure nella definizione circolare del "numero naturale infinito" $\infty = \texttt{succ}(\infty)$, dove $\texttt{succ}(\cdot)$ indica l'operazione di successore.

Il problema dell'equivalenza di strutture non ben fondate, come quelle degli esempi appena visti, potrebbe essere preso come filo conduttore di una parte consistente della ricerca teorica dell'ultimo mezzo secolo, che include alcuni importanti contributi italiani. Vediamone i momenti principali.

La teoria dei domini sviluppata come base matematica della semantica denotazionale consente di interpretare le strutture non ben fondate come soluzioni di equazioni ricorsive, osservando che le circolarità presenti nelle loro definizioni si possono risolvere sfruttando l'esistenza di punti fissi per le funzioni continue di un dominio in sè. Questa soluzione ha l'inconveniente di essere insensibile alla parzialità delle soluzioni ottenute: tipicamente, l'equazione $x = x$ ha sempre soluzione in un dominio D, cioè il minimo punto fisso della funzione identità su D, che è l'elemento minimo $\perp$ di D.[7]

In alternativa alla teoria degli elementi parziali costituita dalla teoria dei domini, si può pensare di risolvere le circolarità presenti nelle definizioni di strutture infinite formalizzandole come elementi di spazi metrici completi: si può allora utilizzare un teorema di Banach sull'esistenza ed unicità di punti fissi per le contrazioni su tali spazi, cioè quelle funzioni $f : E \to E$ definite su spazi metrici completi (E, d) tali che per ogni coppia di punti $x, y \in E$, $d(f(x), f(y)) \leq c \cdot d(x, y)$ per qualche numero reale positivo $c < 1$. Questa strada è stata proposta da Nivat (Arnold e Nivat, 1980) per studiare i termini

[7] Queste problematiche sono state trattate in una fase molto precoce dello sviluppo della teoria dei domini, si veda per esempio (Scott, 1971).

infiniti su una segnatura (del prim'ordine) Σ necessari per la semantica degli schemi di programmi ricorsivi, e poi ripresa proprio nel contesto della semantica dei processi concorrenti in (de Bakker e Zucker, 1982), con spazi metrici completi della forma $P = \{p_0\} \cup \mathcal{P}_c(Act \times P)$, dove si indica con $\mathcal{P}_c(X)$ l'insieme dei sottospazi chiusi di X.[8]

Un approccio ancora diverso alla circolarità proviene dalla teoria degli insiemi non ben fondati (Aczel, 1988), dove l'assioma di fondazione di ZF viene rimpiazzato da assiomi che garantiscono l'esistenza di insiemi come $\Omega = \{\Omega\}$ che inducono circolarità nella relazione di appartenenza: in questo caso, abbiamo che $\Omega \in \Omega$. Tra questi, l'assioma AFA, riscoperto da Aczel ma studiato da Forti e Honsell già nel 1983 (Forti e Honsell, 1983) sviluppando alcune intuizioni di Ennio de Giorgi sui "princìpi di libera costruzione" di insiemi, permette lo sviluppo di una elegante teoria in cui l'uguaglianza tra due insiemi non ben fondati x e y è determinata dall'esistenza di una bisimulazione forte tra i due, che in questo contesto si specializza ad una relazione R sull'universo tale che

$$x' \in x \Rightarrow \exists y'(y' \in y \wedge x' \; R \; y'),$$
$$y' \in y \Rightarrow \exists x'(x' \in x \wedge x' \; R \; y').$$

È così che gli insiemi non ben fondati diventano l'ambiente di una teoria semantica per i termini di CCS (nella versione sincrona descritta in (Milner, 1983)), e la circolarità presente negli insiemi non ben fondati viene utilizzata per interpretare la circolarità coinvolta dalla definizione di processi infiniti come $P = a.P$, associando ad un processo generico P un insieme $[\![P]\!] = \{\langle a, [\![Q]\!]\rangle \mid P \xrightarrow{a} Q\}$. Allora, ad esempio, il termine P viene interpretato come l'insieme (non ben fondato) $[\![P]\!] = \{\langle a, [\![P]\!]\rangle\}$ (si veda (Turi e Rutten, 1998)).

Il problema dell'esistenza di strutture circolari, come si è suggerito, è un problema relativo all'esistenza di soluzioni di equazioni. Consideriamo sistemi *piatti* di equazioni della forma $\vec{x} = a_{\vec{x}}$, dove $\vec{x}$ è una sequenza di indeterminate tratte da un insieme X e $a_{\vec{x}}$ è un insieme i cui elementi sono *atomi* o elementi di X: una formulazione dell'assioma AFA consiste nel richiedere che ogni tale sistema abbia un'unica soluzione (Barwise e Moss, 1996). Si può generalizzare questo assetto, per esempio ottenendo termini infiniti su una segnatura Σ come soluzioni di sistemi di equazioni piatte della forma $x_i = \sigma(x_1, \ldots, x_n)$, dove $\sigma \in \Sigma$ è un operatore di arietà n e $x_1, \ldots, x_n$ sono indeterminate.

[8] La letteratura rilevante in questo ambito è piuttosto estesa, e comprende almeno (Rutten, 1996; Alessi *e altri*, 1995; Bonsangue *e altri*, 1998).

Entrambe queste interpretazioni delle equazioni che definiscono strutture circolari rientrano nel caso generale di una *coalgebra* per un funtore. In generale, se $\mathbb{C}$ è una categoria e $T : \mathbb{C} \longrightarrow \mathbb{C}$ è un funtore, una *T-coalgebra* consiste di un oggetto A di $\mathbb{C}$ con un morfismo $a : A \longrightarrow TA$ in $\mathbb{C}$. Un *omomorfismo* dalla coalgebra $a : A \longrightarrow TA$ alla coalgebra $a' : A' \longrightarrow TA'$ è un morfismo $h : A \longrightarrow A'$ di $\mathbb{C}$ che rende commutativo il diagramma

$$\begin{array}{ccc} A & \xrightarrow{a} & TA \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle Th} \\ A' & \xrightarrow{a'} & TA' \end{array}$$

La T-coalgebra finale (quando esiste) è l'oggetto terminale della categoria delle T-coalgebre. Tra gli esempi di coalgebre finali di opportuni funtori, abbiamo:

- l'insieme dei termini infiniti su una segnatura Σ, in particolare l'insieme dei numeri naturali estesi con un elemento $\infty = \mathtt{succ}(\infty)$;
- l'universo $V_{AFA} = \mathcal{P}_S(V_{AFA})$, dove $\mathcal{P}_S(X)$ è la classe dei sottoinsiemi della classe X, che è modello della teoria assiomatica ZF in cui l'assioma di fondazione è sostituito da AFA; ed infine
- l'insieme dei comportamenti $B = \mathcal{P}_S(Act \times B)$ che interpretano i processi CCS che sono stati il nostro punto di partenza in questa digressione sugli oggetti infiniti.

C'è una evidente dualità (anche terminologica) tra coalgebre finali e algebre iniziali. Queste ultime forniscono un contesto sistematico per la formulazione della semantica dei linguaggi di programmazione (Goguen *e altri*, 1977) riproducendo in forma generale la situazione del teorema di definizione per ricorsione su numeri naturali di Dedekind (Henkin, 1960): data una struttura $\mathfrak{A} = \langle A, f : A \longrightarrow A, a \in A\rangle$ esiste un unico omomorfismo h da $\langle \mathbb{N}, \mathtt{succ}, 0\rangle$ verso $\mathfrak{A}$, cioè un'unica funzione $h : \mathbb{N} \longrightarrow A$ tale che

$$\begin{aligned} h(0) &= a \\ h(\mathtt{succ}(n)) &= f(h(n)). \end{aligned}$$

Questo teorema asserisce che $\langle \mathbb{N}, \mathtt{succ}, 0\rangle$ è l'algebra iniziale per il funtore T : **Set** $\longrightarrow$ **Set** definito da $TX = 1 + X$, perché una T-algebra

$\alpha : TA \longrightarrow A$ è allora completamente determinata dalla scelta di un elemento di A e di una funzione $f : A \longrightarrow A$. Le equazioni soddisfatte dall'omomorfismo h costituiscono uno schema di iterazione, e si può pensare di giustificare analoghi schemi di ricorsione per funzioni utilizzando soltanto l'inizialità del loro dominio. Analogamente, si può interpretare l'unico omomorfismo di coalgebre verso una coalgebra finale come uno schema di coiterazione, utile per definire funzioni che restituiscono strutture infinite. Così, ad esempio, lo stream `ones` $=$ `1 : ones` può essere visto come il risultato di una definizione coiterativa giustificata dal fatto che gli stream (di numeri naturali) costituiscono la T-coalgebra finale, dove $TX = \mathbb{N} \times X$ e l'equazione $x = 1 : x$ può essere vista come una coalgebra dello stesso tipo. Esiste uno stile di programmazione funzionale basato sulla definizione di programmi attraverso schemi di ricorsione associati a tipi di dati (Bird e De Moor, 1997), che recentemente è stato esteso alla programmazione su strutture infinite utilizzando la caratterizzazione di queste come coalgebre finali (Uustalu e Vene, 1999; Lenisa, 1999; Cancila *e altri*, 2003).

Nel contesto coalgebrico, la tecnica privilegiata per la dimostrazione di proprietà è la coinduzione, che sfrutta la proprietà di *estensionalità forte* per cui l'uguaglianza equivale all'esistenza di una bisimulazione, dove la nozione di bisimulazione è parametrizzata dal funtore T appropriato. Abbiamo così la realizzazione un principio di *full abstraction* interna del tutto adeguato al paradigma osservazionale di cui abbiamo brevemente esaminato la formazione, ed al contempo un contesto ideale per la trattazione dei problemi relativi alle strutture circolari.[9]

2.1.4 Tipi e interazione

La nozione di tipo costituisce uno dei canali privilegiati per il passaggio di teorie, tecniche e risultati dalla logica all'informatica, non solo teorica. Ci sono almeno due aspetti attraverso i quali la sua importanza si manifesta anche in una concezione del calcolo come interazione tra agenti computazionali, che sembra essere assente dalle concezioni dei tipi derivate dalla teoria della dimostrazione o dal λ-calcolo. Da un lato, abbiamo le proposte di estensione dell'utilizzo dei tipi a varianti del π-calcolo di Milner (1989), che è ormai diventato il punto di riferimento linguistico per la trattazione di problemi relativi alla comunicazione tra agenti mobili attraverso una radicale riformulazione delle primitive di comunicazione utilizzate in CCS. Ciò

[9] Maggiori dettagli tecnici si trovano nella interessante rassegna di Daniele Turi e Jan Rutten (1998) su domini, spazi metrici e insiemi non ben fondati in una prospettiva coalgebrica.

che viene comunicato tra agenti nel π-calcolo sono, essenzialmente, nomi che si riferiscono a (locazioni di) dati o a canali. È naturale allora imporre restrizioni sul tipo dei valori che possono essere comunicati attraverso uno specifico canale, eventualmente con dei qualificatori per indicare la direzione del flusso dei dati lungo il canale (input, output o entrambe le direzioni) (Pierce e Sangiorgi, 1996), con relative nozioni di inclusione tra tipi (Gay e Hole, 2005). Più recentemente, si è affermata l'utilità di annotare i nomi di canali con espressioni che descrivono la traccia delle possibili interazioni che avvengono tramite il canale in questione: si hanno allora i *tipi sessione* (Honda, 1993; Castagna *e altri*, 2009b; Dezani-Ciancaglini e De' Liguoro, 2010) e loro varianti (Laneve e Padovani, 2008; Castagna *e altri*, 2009a), in cui gli insiemi di tipi sono frammenti di calcoli di processi utilizzati per la descrizione dei comportamenti comunicativi in cui i canali sono coinvolti. In questo modo diventano diagnosticabili mediante analisi statica eventuali anomalie della comunicazione, come deadlock o difetti in meccanismi di delegazione implementati mediante comunicazione di canali. In relazione alla logica, Wadler (Wadler, 2014) ha mostrato una stretta corrispondenza tra tipi sessione e formule della logica lineare.

Anche nell'ambito di quella vasta area di ricerca centrata intorno alla corrispondenza dei tipi come proposizioni, strettamente legata alla teoria della dimostrazione di logiche costruttive, emerge l'interesse di una concezione interattiva del calcolo che si riflette nel modo di intendere i tipi e la loro interpretazione. Questo interesse ha origine nell'ambito della logica lineare (Girard, 1987), da cui trae sia motivazioni che tecniche. È in questo ambito che inizia ad emergere un altro asse della nostra esplorazione delle contaminazioni[10] tra logica e informatica: quello geometrico.

Un tipo può essere considerato come un'interfaccia tra un'espressione e ed il suo contesto π: la loro interazione dà luogo ad una relazione di *ortogonalità* tra espressioni e contesti. Per esempio, nell'ambito generale del λ-calcolo parametrico (Ronchi Della Rocca e Paolini, 2004), in cui si possono studiare uniformemente nozioni diverse di riduzione di λ-termini, usando la notazione $e \Downarrow$ per indicare che la valutazione dell'espressione e termina con un valore (Paolini, 2004) abbiamo:

$$e \perp \pi \iff \pi[e] \Downarrow .$$

Mediante ortogonalità, come in (Danos e Krivine, 2000; Melliès e Vouillon, 2005) si può poi definire una interpretazione dei tipi suggerita dalla generalizzazione della tecnica dei candidati di riducibilità di (Girard, 1972)

[10] Nel senso della critica testuale, non quello epidemiologico.

utilizzata in un teorema di normalizzazione forte per un sistema di *proof-nets* (Girard, 1987, §§4.26.2-4.26.5) dove, per $t : A, u : A^{\perp}$

$$t \perp u \iff \mathbf{CUT}(t, u) \text{ è fortemente normalizzabile.}$$

Questa idea di ortogonalità trova ulteriore generalizzazione in ludica (Girard, 2001) dove, per *disegni* P ed N di opposta polarità ed usando la notazione di (Terui, 2011, Def. 3.7):

$$P \perp N \iff P[N/x] \Downarrow \maltese$$

È a questo livello di generalità che il paradigma osservazionale si riconnette, in modo formale, alla sua origine dalla teoria degli automi: per un automa finito deterministico M ed una parola w (Terui, 2011)

$$M \text{ accetta } w \iff P_M \perp N_w$$

dopo opportuna codifica di M e w come opportuni disegni P_M ed N_w, rispettivamente.

2.2 Composizione di moduli

La regola del taglio nel calcolo dei sequenti per la logica intuizionistica è *asimmetrica* nei ruoli a destra e a sinistra di $\vdash$: in base alla corrispondenza di Curry-de Bruijn-Scott-Läuchli-Lawvere-Howard[11] tra derivazioni e termini:

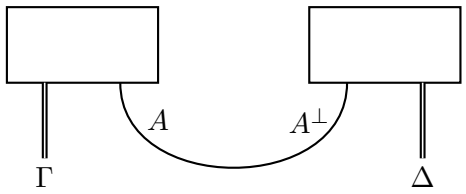
$$\frac{\Gamma \vdash t : A \quad x : A, \Delta \vdash u : B}{\Gamma, \Delta \vdash u[t/x] : B}$$

Questo comporta limitazioni di tipo *geometrico* se pensiamo ai tipi come a *interfacce* per sistemi il cui comportamento è descritto dai termini, i quali hanno essenzialmente una struttura ad albero. Uno dei contributi generali della logica lineare è stato quello di estendere questa forma di composizione a moduli in cui la distinzione tra ingressi ed uscite, asimmetrica nel caso tradizionale, è segnalata dall'applicazione di una operazione di negazione involutiva $(\quad)^{\perp}$ che rende perfettamente simmetrico il ruolo di ingressi ed uscite di un modulo. La composizione è allora descritta dalla versione della regola del taglio per la logica lineare classica (Girard, 1989):

$$\frac{\vdash \Gamma, A \quad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}$$

o, in termini grafici,

[11] L'attribuzione a sei nomi di questa fondamentale scoperta ha una storia complicata, per la quale si rimanda a (Cardone e Hindley, 2009, §8).



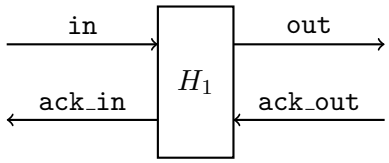
Questa nozione di composizione è centrale nella concezione dei tipi per processi che è alla base delle *interaction categories* di Abramsky (Abramsky *e altri*, 1996), dalla quale proviene l'idea, derivata dall'algebra dei processi (Hoare, 1985), di identificare la composizione di processi con la loro composizione parallela nascondendo all'ambiente esterno il canale attraverso cui la comunicazione ha luogo. Vediamo brevemente come questa idea si sviluppa fino a formare un paradigma di definizione che pervade gli sviluppi recenti dell'approccio interattivo al calcolo.

Immaginiamo di avere un componente hardware H_1 il cui comportamento è descritto dalla seguente specifica:

- H_1 riceve un segnale in ingresso sul canale `in`,
- emette un segnale sul canale `out`,
- aspetta un segnale sul canale `ack_out` e, avendolo ricevuto,
- emette un segnale sul canale `ack_in`.

Il comportamento di H_1 è conforme ad un protocollo di comunicazione di tipo *handshake*, in cui ci sono due tipi di segnali (di tipo *request* e di tipo *acknowledge*), ciascuno dei quali in *input* o in *output*, che si alternano (su ciascun lato del componente) iniziando con un segnale di tipo *request*.

L'interfaccia del componente H_1 può essere rappresentata graficamente dallo schema seguente:



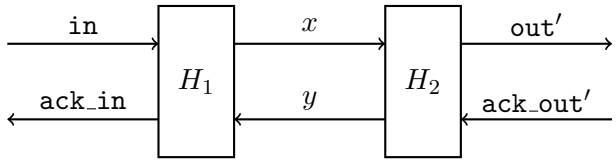
Per descrivere i possibili comportamenti del componente H_1 usiamo *tracce* (Hoare, 1985), parole sull'alfabeto $\alpha H_1 = \{\texttt{in}, \texttt{ack\_in}, \texttt{out}, \texttt{ack\_out}\}$:

$$\tau H_1 = \{\varepsilon, \texttt{in}, \texttt{in} \cdot \texttt{out}, \texttt{in} \cdot \texttt{out} \cdot \texttt{ack\_out}, \texttt{in} \cdot \texttt{out} \cdot \texttt{ack\_out} \cdot \texttt{ack\_in}, \ldots\}$$

Aggiungiamo un secondo elemento H_2 sull'alfabeto

$$\alpha H_2 = \{\texttt{in}', \texttt{ack\_in}', \texttt{out}', \texttt{ack\_out}'\}.$$

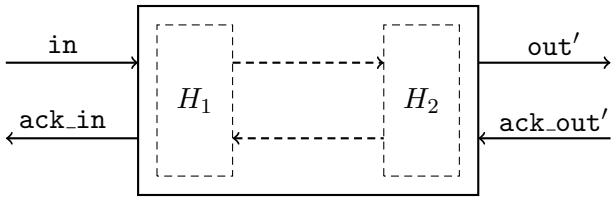
La composizione avviene rinominando i segnali `out` e $\texttt{in}'$ come x, `ack_out` e $\texttt{ack\_in}'$ come y. Abbiamo così il sistema $H_1 \| H_2$:



il cui comportamento è definito dall'insieme di tracce sull'alfabeto $\alpha(H_1 \| H_2) = \alpha H_1 \cup \alpha H_2$:

$$\begin{aligned}
\tau(H_1 \| H_2) = \quad & \{w \in \alpha(H_1 \| H_2)^* \mid w \upharpoonright \alpha H_1 \in \tau H_1, w \upharpoonright \alpha H_2 \in \tau H_2\} \\
= \{ \quad & \\
& \varepsilon, \\
& \texttt{in}, \\
& \texttt{in} \cdot x, \\
& \texttt{in} \cdot x \cdot \texttt{out}', \\
& \texttt{in} \cdot x \cdot \texttt{out}' \cdot \texttt{ack_out}', \\
& \texttt{in} \cdot x \cdot \texttt{out}' \cdot \texttt{ack_out}' \cdot y, \\
& \texttt{in} \cdot x \cdot \texttt{out}' \cdot \texttt{ack_out}' \cdot y \cdot \texttt{ack_in}, \ldots \\
& \}
\end{aligned}$$

Si possono nascondere agli osservatori esterni i canali x ed y, mediante l'operazione di *hiding*, ottenendo così il sistema finale $[H_1 \| H_2]$



le cui tracce hanno la forma

$$\begin{aligned}\tau([H_1 \| H_2]) = \{ & \\ & \varepsilon, \\ & \texttt{in}, \\ & \texttt{in} \cdot \texttt{out}', \\ & \texttt{in} \cdot \texttt{out}' \cdot \texttt{ack\_out}', \\ & \texttt{in} \cdot \texttt{out}' \cdot \texttt{ack\_out}' \cdot \texttt{ack\_in}, \ldots \\ \}\end{aligned}$$

2.3 Dialoghi

La stessa alternanza di interazioni tra un *Sistema* ed un *Ambiente*, con la prima mossa all'Ambiente, si può riformulare in termini di *dialoghi*, formalizzati come *giochi* tra un Proponente ed un Opponente le cui strategie sono programmi interattivi (Coquand, 1995). Il cambio di polarità Sistema/Ambiente è evidente nella composizione di strategie, che è basata sulla stessa idea (*parallel composition + hiding*) della composizione di componenti handshake utilizzata prima per definire il sistema $[H_1 \| H_2]$. Questa interpretazione dell'interazione tra agenti computazionali riproduce l'assetto sul quale è basata la semantica dialogica per la logica intuizionistica sviluppata da Paul Lorenzen, a partire da (Lorenzen, 1961), in cui le mosse sono attacchi e difese.

In altro contesto, nella sua revisione della teoria dei funzionali ricorsivi, Kleene (1985) aveva introdotto un modello di calcolo che comportava scambi di domande e risposte con oracoli, le cui analogie con la teoria degli algoritmi sequenziali di Berry e Curien (1982) è stata analizzata da Antonio Bucciarelli (1994). Questo modello di calcolo di Kleene ha poi ispirato il lavoro (non pubblicato) di Robin Gandy e Giovanni Pani su una interpretazione dialogica della calcolabilità adatta alla semantica del linguaggio PCF, che culmina nella formulazione della *game semantics* utilizzata (in diverse versioni) nella soluzione del problema della full abstraction per questo linguaggio in (Abramsky *e altri*, 2000) e (Hyland e Ong, 2000).

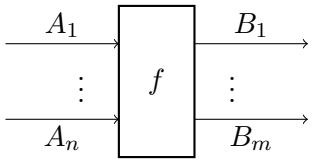
In ambito più strettamente logico, la situazione tipica è quella di un dibattito tra Proponente ed Opponente intorno alla verità di una formula in forma prenessa, per esempio dell'aritmetica elementare. ∀belardo, che cerca un controesempio alla verità della formula, muove quando la formula è della forma $\forall i\ \varphi(i)$ scegliendo i_0. La mossa del Proponente, ∃loisa, che difende la verità della formula contro ∀belardo, consiste nella scelta, quando

la formula è della forma $\exists i\ \varphi(i)$, di un i_0. In entrambi i casi il gioco continua con la formula $\varphi(i_0)$, e termina quando la formula è atomica: se è falsa vince $\forall$belardo, altrimenti è $\exists$loisa a vincere. Nel caso in cui la formula è in forma prenessa, si dimostra che $\exists$loisa ha una strategia vincente nel gioco associato ad una formula φ esattamente quando la formula φ è intuizionisticamente vera. La verità classica corrisponde ad una variante del gioco in cui $\exists$loisa, e solo lei, è autorizzata a ritornare ad una posizione precedente, modificando quindi la sua scelta di i_0 (Coquand, 1995). Estendendo questa possibilità di *backtracking* a tutti i giocatori, si ottengono giochi in cui i giocatori imparano, migliorando le loro mosse procedendo per tentativi ed errori: le strategie ricorsive in questo gioco caratterizzano il contenuto costruttivo di sottosistemi dell'aritmetica classica (Berardi *e altri*, 2010), interpretando le dimostrazioni classiche come strategie interattive di apprendimento (Berardi e de' Liguoro, 2009, 2012).

3 Geometria

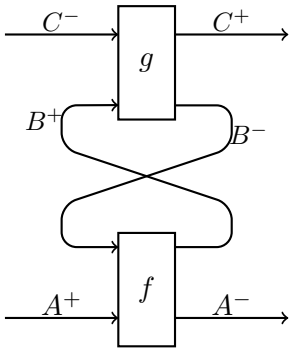
La definizione della composizione di strategie per i giochi dialogici (o di componenti handshake) esibisce una forma caratteristica che ha origine dagli sviluppi formali della *geometria dell'interazione* di Girard (1988). Questo programma di descrizione algebrica della dinamica delle dimostrazioni in logica lineare ha subito trovato applicazione nell'ambito dei linguaggi di programmazione dopo la scoperta di un'implementazione della riduzione ottimale per il λ-calcolo di Lévy da parte di Lamping (1990). La relazione dell'algoritmo di Lamping con la geometria dell'interazione, osservata in (Gonthier *e altri*, 1992), ha dato origine ad una esplorazione sistematica delle proprietà dei *sharing graphs* utilizzati da Lamping da parte di Asperti, Laneve, Guerrini, Martini, Masini (Asperti e Guerrini, 1998).

La geometria dell'interazione ammette una formulazione astratta (Abramsky *e altri*, 2002) attraverso categorie monoidali, in cui un morfismo $f : A_1 \otimes \cdots \otimes A_n \longrightarrow B_1 \otimes \cdots \otimes B_m$ può essere presentato mediante un diagramma (Joyal e Street, 1991) della forma



In particolare, questa formulazione utilizza categorie monoidali in cui, ad ogni morfismo $f : A \otimes U \longrightarrow B \otimes U$ è associata la sua *traccia* $\mathrm{Tr}(f)$:

$A \longrightarrow B$, che si può interpretare genericamente come una forma di feedback, o iterazione (Joyal *e altri*, 1996). In questo contesto la composizione parallela con *hiding* di due morfismi in una categoria monoidale con traccia $\mathbb{C}$, $f : A^+ \otimes B^- \longrightarrow A^- \otimes B^+$ e $g : B^+ \otimes C^- \longrightarrow B^- \otimes C^+$ assume la seguente forma caratteristica:

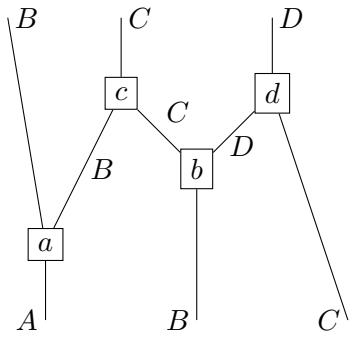


L'utilizzo del feedback rappresentato dalla traccia permette dunque di definire la composizione di morfismi di una nuova categoria, Int($\mathbb{C}$). La categoria Int($\mathbb{C}$) generalizza la costruzione del gruppo delle differenze di un monoide commutativo con cancellazione, esemplificata dalla costruzione dei numeri interi a partire dal monoide additivo dei numeri naturali. Quando applicata a strategie f, g, lo stesso diagramma esprime l'interazione tra le due, dove A^+ rappresenta il tipo delle mosse del Proponente (Sistema) ed A^- quello delle mosse dell'Opponente (Ambiente) (Abramsky, 1996). Nel contesto originale della Geometria dell'Interazione di Girard (1988), questa operazione di composizione diventa la formula di esecuzione, che rappresenta la dinamica dell'eliminazione del taglio in una dimostrazione nel calcolo dei sequenti (Haghverdi e Scott, 2011).

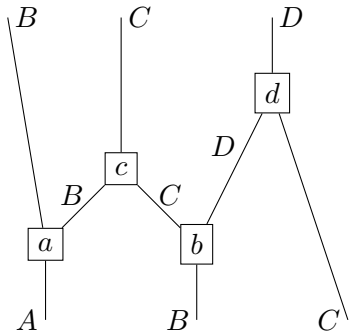
3.1 Diagrammi e automi concorrenti

Nel lavoro originale di Joyal e Street sull'interpretazione geometrica delle categorie monoidali (Joyal e Street, 1991), i morfismi e la loro composizione sono rappresentati da diagrammi.

Per esempio, il morfismo $(aBC); (BBbC); (Bcd)$ è descritto dal diagramma



mentre $(abC); (BcDC); (BCd)$ è descritto da



Si può dimostrare (Joyal e Street, 1991, Th. 1.2) che diagrammi (interpretati come spazi topologici: *progressive plane diagrams*) ottenuti per deformazioni continue di uno stesso diagramma hanno la stessa valutazione in una categoria monoidale. Analoghi risultati si ottengono per altre classi di diagrammi, associati a categorie monoidali con struttura addizionale, in particolare, come si è visto, categorie con traccia o compattamente chiuse come le categorie della forma $\mathrm{Int}(\mathbb{C})$: le manipolazioni equazionali in queste categorie possono essere sostituite da manipolazioni geometriche (Selinger, 2011). Oltre al loro intrinseco interesse concettuale, queste caratterizzazioni permettono approcci geometrici negli ambiti applicativi in cui trovano impiego le corrispondenti categorie, come le versioni categoriali della meccanica quantistica (Coecke, 2010), i modelli basati sui giochi (Melliès, 2012), i modelli di calcoli funzionali reversibili e conservativi (Lafont, 2003; James e Sabry, 2012).

La descrizione geometrica dei morfismi di categorie monoidali ha un impiego naturale anche nell'ambito della teoria della *concorrenza*, fenomeno caratteristico di sistemi di calcolo distribuiti ma originariamente isolato in sede teorica proprio nel contesto di quel particolare formalismo grafico costituito dalle reti di Petri, in cui è possibile specificare relazioni di dipendenza

causale tra eventi in un sistema. Una particolare connessione tra reti di Petri e categorie monoidali è stata al centro di una serie di lavori di Ugo Montanari e suoi collaboratori, in particolare José Meseguer (Meseguer e Montanari, 1990). Nella stessa area di ricerca è emersa anche la possibilità di rappresentare la concorrenza mediante tecniche derivate dalla teoria delle categorie di dimensione superiore. Il *tile model* sviluppato da Montanari con Fabio Gadducci (Gadducci e Montanari, 2000) utilizza le categorie doppie per la formalizzazione di un modello di CCS in cui è rappresentata fedelmente la concorrenza di transizioni, a differenza di quanto accade con formalismi tradizionali per l'equivalenza osservazionale di processi.

Un'altra via attraverso la quale la rilevanza di categorie di ordine superiore emerge nella costruzione di modelli di processi concorrenti è l'osservazione che l'indipendenza causale di eventi permette la loro manifestazione in qualsiasi ordine temporale. Stark (1989) ha usato la nozione di residuo tratta dalla teoria della riduzione per definire una congruenza sulle computazioni, a meno di permutazioni di transizioni causalmente indipendenti, seguendo le tracce del lavoro di Lévy (1978) per la riduzione nel λ-calcolo. Si ottiene così una nozione di *automa concorrente* che permette la manifestazione, allo stesso passo di calcolo, di un insieme di transizioni causalmente indipendenti identificando tutte le possibili sequenzializzazioni della loro esecuzione. L'idea di usare direttamente transizioni n-dimensionali ($n > 1$) per rappresentare l'esecuzione in un solo passo di n eventi è di (Pratt, 1991), in un formalismo ispirato dalla teoria delle n-categorie, poi semplificato da van Glabbeek (2006) e da Cattani e Sassone (1996). Più recentemente, questa concezione geometrica della concorrenza ha motivato lo studio di relazioni di *omotopia* tra cammini orientati (Grandis, 2009; Fajstrup *e altri*, 2006).

4 Osservazioni conclusive

Anche se abbiamo scelto di restringere la nostra attenzione a quelle interazioni tra logica e informatica che hanno maggiore rilievo per lo sviluppo dei fondamenti di una nuova concezione del calcolo sulla base del bagaglio di tecniche e nozioni sviluppate attraverso la ricerca in logica, la nostra esposizione è ben lontana dall'esaurire tutte le direzioni in cui queste interazioni hanno avuto luogo. Così, niente è stato detto dell'utilizzo di logiche modali per la conoscenza, in cui si possono formulare proprietà della nozione di conoscenza comune che si sono rivelate essenziali nel dimostrare risultati negativi relativi al coordinamento di agenti in sistemi distribuiti con comunicazione asincrona o in cui sia possibile la perdita di messaggi (Halpern e

Moses, 1990; Chandy e Misra, 1986). Un altro ambito di applicazioni è costituito dall'elaborazione di modelli formali di sistemi di agenti, per esempio logiche per la formalizzazione di contratti relativi a servizi web (Bartoletti e Zunino, 2010), ambito in cui emergono anche problematiche relative ad autenticazione ed autorizzazione in sistemi distribuiti (Abadi *e altri*, 1993; Li *e altri*, 2003).[12]

Anche alcuni temi direttamente collegati con le nozioni generali che abbiamo utilizzato nella nostra narrazione sono stati solo evocati. La teoria strutturale della dimostrazione è stata costantemente sullo sfondo delle nostre considerazioni. Una nozione di interazione di portata assai generale è stata elaborata nel suo ambito attraverso la ricerca in logica lineare, iniziata in Italia ancor prima che venisse pubblicato l'articolo originale di Girard (Girard, 1987), contribuendo all'apertura di direzioni di ricerca che ormai sono parte del folklore dell'argomento, dalle connessioni con le reti di Petri (Asperti, 1987) alla logica lineare non commutativa con le sue applicazioni alle grammatiche categoriali nella formalizzazione di Lambek (Abrusci e Ruet, 1999).[13]

È indimenticabile l'influsso della Teoria dei Tipi intuizionistica elaborata da Per Martin-Löf (1984) con una chiara percezione delle sue applicazioni informatiche, come linguaggio di programmazione in cui i programmi sono dimostrazioni delle proposizioni che esprimono le loro specifiche (Martin-Löf, 1982). Su un versante più filosofico, le lezioni di Siena di Martin-Löf (1985) hanno sviluppato una teoria del significato delle costanti logiche costruttive che da un lato, accanto alle proposte di Dummett e Prawitz, ha costituito un punto di riferimento nella filosofia della logica per le successive elaborazioni semantiche (Moriconi, 1984);[14] d'altro lato, evidenziando la rilevanza della nozione di giudizio per una corretta formulazione delle regole logiche, ha ispirato l'architettura del *logical framework* sviluppato in (Harper *e altri*,

[12] La complessa rete di concetti deontici implicita nell'implementazione di contratti sul web è ovviamente collegata con l'oggetto del contributo di Giovanni Sartor al presente volume.

[13] I membri del gruppo di Logica e Geometria della Cognizione di Roma Tre hanno molto contribuito a definire l'immagine di quest'area. Tra l'altro, la monografia di Girard (Girard, 2011) al culmine degli ultimi trent'anni della sua ricerca ha preso forma attraverso una serie di lezioni tenute nel 2004 all'interno delle attività di questo gruppo. La teoria della dimostrazione strutturale ed i suoi risvolti tecnici nell'ambito della complessità computazionale implicita sono approfonditi nel capitolo di Simone Martini, nel presente volume.

[14] Per avere un'idea dei recenti progressi in quest'area si può consultare il numero 148(3) del Febbraio 2006 della rivista *Synthese*, interamente dedicato alla *proof-theoretic semantics*.

1993) per l'implementazione di logiche generali.

Infine, riconosciamo la totale assenza, dalle nostre considerazioni, della programmazione logica e della vasta area della deduzione automatica, in cui importanti contributi provengono proprio dal nostro Paese. È un'assenza motivata dall'impostazione generale della nostra presentazione, ma anche da una versione personalizzata dell'ultima proposizione del *Tractatus* di Wittgenstein: di ciò di cui non si sa parlare si deve tacere.

Ringraziamenti. Questo testo conterrebbe molti più errori ed omissioni senza l'amichevole aiuto di Stefano Berardi, Luca Padovani, Luca Paolini e Luca Roversi, che ringrazio per la pazienza.

Riferimenti bibliografici

Abadi M.; Burrows M.; Lampson B.; Plotkin G. (1993). A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, **15**(4), 706–734.

Abramsky S. (1996). Retracing some paths in process algebra In *CONCUR '96: Concurrency Theory, 7th International Conference*. A cura di Montanari U., Sassone V., Lecture Notes in Computer Science, pp. 1–17. Springer-Verlag.

Abramsky S.; Gay S.; Nagarajan R. (1996). Interaction categories and the foundations of typed concurrent programming. In *Proceedings of the NATO Advanced Study Institute on Deductive Program Design*, NATO ASI Series F, pp. 35–113. Springer-Verlag.

Abramsky S.; Jagadeesan R.; Malacaria P. (2000). Full abstraction for pcf. *Information and Computation*, pp. 409–470.

Abramsky S.; Haghverdi E.; Scott P. (2002). Geometry of interaction and linear combinatory algebras. In *Mathematical Structures in Computer Science*, volume 12, pp. 625–665.

Abrusci V. M.; Ruet P. (1999). Non-commutative logic I: The multiplicative fragment. *Annals of Pure and Applied Logic*, **101**(1), 29–64.

Aczel P. (1988). *Non-well-founded Sets* in Lecture Notes. Numero 14. Center for the Study of Language and Information, Stanford University.

Alessi F.; Baldan P.; Bellè G.; Rutten J. (1995). Solutions of functorial and non-functorial metric domain equations. *Electronic Notes in Theoretical Computer Science*, **1**, 1–12. {MFPS} XI, Mathematical Foundations of Programming Semantics, Eleventh Annual Conference.

Amadio R.; Curien P.-L. (1998). *Domains and Lambda-Calculi*, volume 46 di *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, England.

Arnold A.; Nivat M. (1980). The metric space of infinite trees, algebraic and topological properties. *Fundamenta Informaticæ*, **III**(4), 445–476.

Asperti A. (1987). A logic for concurrency. Technical Report, Dipartimento di Informatica, Università di Pisa.

Asperti A.; Guerrini S. (1998). *The Optimal Implementation of Functional Programming Languages*. Cambridge University Press, New York, NY, USA.

Awodey S. (2012). Type theory and homotopy In *Epistemology versus Ontology - Essays on the Philosophy and Foundations of Mathematics in Honour of Per Martin-Löf*. A cura di Dybjer P., Lindström S., Palmgren E., Sundholm G., volume 27 di *Logic, Epistemology, and the Unity of Science*, pp. 183–201. Springer-Verlag.

Baez J. C.; Lauda A. D. (2011). A prehistory of n-categorical physics. In *Deep beauty. Understanding the quantum world through mathematical innovation. Papers based on the presentations at the deep beauty symposium, Princeton, NJ, USA, October 3–4, 2007*, pp. 13–128. Cambridge: Cambridge University Press.

Bartoletti M.; Zunino R. (2010). A calculus of contracting processes. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pp. 332–341. IEEE Computer Society.

Barwise J.; Moss L. (1996). *Vicious Circles*. Center for the Study of Language and Information.

Berardi S.; de' Liguoro U. (2009). Toward the interpretation of non-constructive reasoning as non-monotonic learning. *Information and Computation*, **207**(1), 63–81.

Berardi S.; de' Liguoro U. (2012). Interactive realizers: A new approach to program extraction from nonconstructive proofs. *ACM Transactions on Computational Logic*, **13**(2), 11.

Berardi S.; Coquand T.; Hayashi S. (2010). Games with 1-backtracking. *Annals of Pure and Applied Logic*, **161**(10), 1254–1269.

Berry G.; Curien P.-L. (1982). Sequential algorithms on concrete data structures. *Theoretical Computer Science*, **20**, 265–321.

Bird R.; De Moor O. (1997). *The Algebra of Programming*. Prentice Hall International Series in Computer Science. Prentice Hall.

Böhm C. (1968). Alcune proprietà delle forme β-η-normali nel λ-K-calcolo. Pubblicazione no. 696, Istituto per le Applicazioni del Calcolo, C.N.R., Roma.

Bonsangue M. M.; van Breugel F.; Rutten J. J. M. M. (1998). Generalized metric spaces: Completion, topology, and power domains via the Yoneda embedding. *Theoretical Computer Science*, **193**(1-2), 1–51.

Bucciarelli A. (1994). Another approach to sequentiality: Kleene's unimonotone functions In *Mathematical Foundations of Programming Semantics, 9th International Conference, New Orleans, LA, USA, April 7-10, 1993, Proceedings*. A cura di Brookes S. D., Main M. G., Melton A., Mislove M. W., Schmidt D. A., volume 802 di *Lecture Notes in Computer Science*, pp. 333–358. Springer.

Burstall R.; Honsell F. (1988). A natural deduction treatment of operational semantics In *Foundations of Software Technology and Theoretical Computer Science*. A cura di Nori K., Kumar S., volume 338 di *Lecture Notes in Computer Science*, pp. 250–269. Springer-Verlag.

Burstall R. M. (1969). Proving properties of programs by structural induction. *Computer Journal*, **12**(1), 41–48.

Cancila D.; Honsell F.; Lenisa M. (2003). Generalized coiteration schemata. *Electronic Notes in Theoretical Computer Science*, **82**(1), 76–93.

Cardelli L. (2008). Bitonal membrane systems: Interactions of biological membranes. *Theoretical Computer Science*, **404**(1-2), 5–18.

Cardone F.; Hindley J. (2009). History of lambda-calculus and combinators In *Handbook of the History of Logic. Volume 5. Logic from Russell to Church*. A cura di Gabbay D., Woods J., pp. 723–817. Elsevier.

Castagna G.; Gesbert N.; Padovani L. (2009a). A Theory of Contracts for Web Services. *ACM Transactions on Programming Languages and Systems*, **31**(5).

Castagna G.; Dezani-Ciancaglini M.; Giachino E.; Padovani L. (2009b). Foundations of session types. In *Proceedings of the 11th ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, pp. 219–230, New York, NY, USA. ACM.

Cattani G. L.; Sassone V. (1996). Higher dimensional transition systems. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pp. 55–62.

Chandy K. M.; Misra J. (1986). How processes learn. *Distributed Computing*, **1**(1), 40–52.

Chomsky N. (1959). On certain formal properties of grammars. *Information and Control*, **2**(2), 137–167.

Church A.; Rosser J. B. (1936). Some properties of conversion. *Transactions of the American Mathematical Society*, **39**, 472–482.

Coecke B. (2010). Quantum picturalism. *Contemporary Physics*, **51**(1), 59–83.

Coecke B.; Duncan R. (2011). Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, **13**(4), 043016.

Coppo M.; Dezani M.; Venneri B. (1981). Functional characters of solvable terms. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, **27**, 45–58.

Coppo M.; Dezani M.; Honsell F.; Longo G. (1983). Applicative information systems In *Colloquium on Trees and Algebra in Programming.* A cura di Ausiello G., Protasi M., volume 159 di *Lecture Notes in Computer Science*, pp. 33–64. Springer-Verlag, Berlin.

Coppo M.; Dezani M.; Honsell F.; Longo G. (1984). Extended type structures and filter lambda models In *Logic Colloquium '82.* A cura di Lolli G., Longo G., Marcja A., pp. 241–262. North-Holland Co., Amsterdam.

Coquand T. (1995). A semantics of evidence for classical arithmetic. *Journal of Symbolic Logic*, **60**(1), 325–337.

Curry H. B.; Feys R. (1958). *Combinatory Logic, Volume I.* North-Holland Co., Amsterdam.

Danos V.; Krivine J.-L. (2000). Disjunctive tautologies as synchronisation schemes In *Computer Science Logic, 14th Annual Conference of the EAC-SL, Fischbachau, Germany, August 21-26, 2000, Proceedings.* A cura di Clote P., Schwichtenberg H., volume 1862 di *Lecture Notes in Computer Science*, pp. 292–301. Springer.

de Bakker J.; Zucker J. (1982). Processes and the denotational semantics of concurrency. *Information and Control*, **54**(1-2), 70–120.

De Nicola R. (2011a). Behavioral equivalences In *Encyclopedia of Parallel Computing.* A cura di Padua D. A., pp. 120–127. Springer.

De Nicola R. (2011b). Process algebras In *Encyclopedia of Parallel Computing.* A cura di Padua D. A., pp. 1624–1636. Springer.

De Nicola R.; Hennessy M. (1984). Testing equivalences for processes. *Theoretical Computer Science*, **34**, 83–133.

Dezani-Ciancaglini M.; De' Liguoro U. (2010). Sessions and session types: An overview. In *Proceedings of the 6th International Conference on Web Services and Formal Methods*, Lecture Notes in Computer Science, pp. 1–28. Springer-Verlag.

Fajstrup L.; Raußen M.; Goubault E. (2006). Algebraic topology and concurrency. *Theoretical Computer Science*, **357**(1-3), 241–278.

Forti M.; Honsell F. (1983). Set theory with free construction principles. *Annali della Scuola Normale Superiore di Pisa, Classe di Scienze*, **10**(4), 493–522.

Gadducci F.; Montanari U. (2000). The tile model In *Proof, Language, and Interaction.* A cura di Plotkin G., Stirling C., Tofte M., pp. 133–166. MIT Press, Cambridge, MA, USA.

Gay S.; Hole M. (2005). Subtyping for session types in the pi calculus. *Acta Informatica*, **42**(2), 191–225.

Giannini P.; Longo G. (1984). Effectively given domains and lambda-calculus models. *Information and Control*, **62**(1), 36–63.

Girard J.-Y. (1972). Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur. Thèse de Doctorat d'État, Univ. Paris VII, Place Jussieu, Paris.

Girard J.-Y. (1987). Linear logic. *Theoretical Computer Science*, **50**, 1–102.

Girard J.-Y. (1988). Geometry of interaction I: an interpretation of system *F* In *Logic Colloquium '88*. A cura di Ferro R., Bonotto C., Valentini S., Zanardo A., pp. 221–260. North-Holland.

Girard J.-Y. (1989). Towards a geometry of interaction In *Categories in Computer Science and Logic: Proc. of the Joint Summer Research Conference*. A cura di Gray J. W., Scedrov A., pp. 69–108. American Mathematical Society, Providence, RI.

Girard J.-Y. (2001). Locus solum. *Mathematical Structures in Computer Science*, **11**, 299–506.

Girard J.-Y. (2011). *The blind spot. Lectures on logic.* Zürich: European Mathematical Society (EMS).

Goguen J.; Thatcher J.; Wagner E.; Wright J. (1977). Initial algebra semantics and continuous algebras. *Journal of the ACM*, **24**, 68–95.

Gonthier G.; Abadi M.; Lévy J.-J. (1992). The geometry of optimal lambda reduction. In *Proceedings of the 19th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 15–26. ACM.

Grandis M. (2009). *Directed algebraic topology : models of non-reversible worlds*. Cambridge University Press, Cambridge.

Guglielmi A. (2007). A system of interaction and structure. *ACM Transactions on Computational Logic*, **8**(1).

Guiraud Y. (2006). The three dimensions of proofs. *Annals of Pure and Applied Logic*, **141**(1-2), 266–295.

Haghverdi E.; Scott P. (2011). Geometry of interaction and the dynamics of proof reduction: A tutorial In *New Structures for Physics*. A cura di Coecke B., volume 813 di *Lecture Notes in Physics*, pp. 357–417. Springer-Verlag.

Halpern J. Y.; Moses Y. (1990). Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, **37**(3), 549–587.

Harper R.; Honsell F.; Plotkin G. D. (1993). A framework for defining logics. *Journal of the ACM*, **40**, 143–184.

Henkin L. (1960). On mathematical induction. *American Mathematical Monthly*, **67**, 323–338.

Hoare C. A. R. (1985). *Communicating Sequential Processes*. Prentice-Hall, Inc.

Honda K. (1993). Types for dyadic interaction. In *CONCUR '93, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 23-26, 1993, Proceedings*, volume 715 di *Lecture Notes in Computer Science*, pp. 509–523. Springer.

Hyland J. M. E.; Ong C.-H. L. (2000). On full abstraction for pcf: I, II, and III. *Information and Computation*, **163**(2), 285–408.

James R. P.; Sabry A. (2012). Information effects In *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*. A cura di Field J., Hicks M., pp. 73–84.

Joyal A.; Street R. (1991). The Geometry of Tensor Calculus, I. *Advances in Mathematics*, **88**, 55–112.

Joyal A.; Street R.; Verity D. (1996). Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, **119**, 447–468.

Kahn G. (1974). The semantics of a simple language for parallel programming. In *Proceedings of the IFIP Congress 74*, pp. 471–475. North-Holland, Amsterdam.

Kleene S. C. (1985). Unimonotone functions of finite types (Recursive functionals and quantifiers of finite types revisited IV) In *Recursion Theory*. A cura di Nerode A., Shore R. A., pp. 119–138. American Mathematical Society, Providence, Rhode Island.

Lafont Y. (2003). Towards an algebraic theory of boolean circuits. *Journal of Pure and Applied Algebra*, **184**(2-3), 257 – 310.

Lamping J. (1990). An algorithm for optimal lambda calculus reduction. In *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 16–30, New York, NY, USA. ACM.

Laneve C.; Padovani L. (2008). The pairing of contracts and session types In *Concurrency, Graphs and Models*. A cura di Degano P., Nicola R., Meseguer J., pp. 681–700. Springer-Verlag.

Leinster T. (2004). *Higher Operads, Higher Categories* in London Mathematical Society Lecture Note Series. Numero 298. Cambridge University Press.

Lenisa M. (1999). From set-theoretic coinduction to coalgebraic coinduction: some results, some problems. *Electronic Notes in Theoretical Computer Science*, **19**, 2–22.

Lévy J.-J. (1978). *Réductions correctes et optimales dans le λ-calcul*. Thèse de Doctorat d'Etat, Univ. Paris VII, Paris.

Li N.; Grosof B. N.; Feigenbaum J. (2003). Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. Inf. Syst. Secur.*, **6**(1), 128–171.

Longo G. (2004). Some topologies for computations. In *Geometrie au XX siècle, 1930-2000*, Paris. Hermann.

Longo G.; Martini S. (1986). Computability in higher types, $P\omega$ and the completeness of type assignment. *Theoretical Computer Science*, **46**(3), 197–217.

Longo G.; Moggi E. (1991). Constructive natural deduction and its 'omega-set' interpretation. *Mathematical Structures in Computer Science*, **1**(2), 215–254.

Lorenzen P. (1961). Ein dialogisches Konstructivitätskriterium. In *Infinitistic Methods*, pp. 193–200. Pergamon Press and PWN, Oxford and Warsaw.

Martin-Löf P. (1982). Constructive mathematics and computer programming In *Logic, Methodology and Philosophy of Science, VI*. A cura di Cohen L. J., Łoś J., Pfeiffer H., Podewski K.-P., pp. 153–175. North-Holland Co., Amsterdam.

Martin-Löf P. (1984). *Intuitionistic Type Theory*. Studies in Proof Theory. Bibliopolis, Napoli, Italy.

Martin-Löf P. (1985). On the meanings of the logical constants and the justifications of the logical laws In *Atti degli Incontri di Logica Matematica*. A cura di Bernardi C., Pagli P., pp. 203–281. Univ. di Siena.

Masini A. (1992). 2-sequent calculus: a proof theory of modalities. *Annals of Pure and Applied Logic*, **58**(3), 229–246.

Melliès P.-A. (2012). Game semantics in string diagrams. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pp. 481–490. IEEE.

Melliès P.-A.; Vouillon J. (2005). Recursive polymorphic types and parametricity in an operational framework. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*, pp. 82–91. IEEE Computer Society.

Meseguer J.; Montanari U. (1990). Petri nets are monoids. *Information and Control*, **88**(2), 105–155.

Milner R. (1975). Processes: a mathematical model of computing agents In *Logic Colloquium '73*. A cura di Rose H., Shepherdson J., pp. 157–174. North-Holland Co., Amsterdam.

Milner R. (1977). Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, **4**, 1–22.

Milner R. (1980). *A Calculus of Communicating Systems*, volume 92 di *Lecture Notes in Computer Science*. Springer.

Milner R. (1983). Calculi for synchrony and asynchrony. *Theoretical Computer Science*, **25**(3), 267–310.

Milner R. (1989). *Communication and Concurrency*. Prentice-Hall, Inc.

Milner R. (1993). Elements of interaction – Turing award lecture. *Commun. ACM*, **36**(1), 78–89.

Moggi E. (1988). Partial morphisms in categories of effective objects. *Information and Computation*, **76**(2-3), 250–277.

Moggi E. (1991). Notions of computation and monads. *Information and Computation*, **93**(1), 55–92.

Moore E. F. (1956). Gedanken-experiments on sequential machines. In *Automata studies*, Annals of mathematics studies, no. 34, pp. 129–153. Princeton University Press, Princeton, N. J.

Moriconi E. (1984). *Dimostrazioni e Significato*. Franco Angeli, Milano, Italy.

Morris J. H. (1968). *Lambda-calculus Models of Programming Languages*. Tesi di Dottorato di Ricerca, Massachusetts Inst. Technology, Cambridge, Mass., U.S.A.

Myhill J. (1957). Finite automata and the representation of events. Technical report WADD TR-57-624, Wright Patterson Air Force Base, Ohio.

Nerode A. (1958). Linear automaton transformations. *Proceedings of the American Mathematical Society*, **9**(4), pp. 541–544.

Newman M. H. A. (1942). On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, **43**(2), 223–243.

Paolini L. (2004). *Lambda-theories: some investigations*. Tesi di Dottorato di Ricerca, Università degli Studi di Genova and Université de la Méditerranée (Aix-Marseille II).

Park D. (1981). Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pp. 167–183. Springer-Verlag.

Pierce B. C.; Sangiorgi D. (1996). Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, **6**(5), 409–453.

Plotkin G. D. (1977). LCF considered as a programming language. *Theoretical Computer Science*, **5**, 223–257.

Plotkin G. D. (2004). The origins of structural operational semantics. *Journal of Logic and Algebraic Programming*, **60-61**, 3–15.

Pratt V. (1991). Modeling concurrency with geometry. In *Proc. 18th Ann. ACM Symposium on Principles of Programming Languages*, pp. 311–322.

Robinson E.; Rosolini G. (1988). Categories of partial maps. *Information and Computation*, **79**(2), 95–130.

Ronchi Della Rocca S.; Paolini L. (2004). *The Parametric λ-Calculus: a Metamodel for Computation*. Texts in Theoretical Computer Science: An EATCS Series. Springer-Verlag, Berlin.

Rosenblueth A.; Wiener N.; Bigelow J. (1943). Behavior, purpose and teleology. *Philosophy of Science*, **10**, 18–24.

Rosolini G. (1986). *Continuity and Effectiveness in Topoi*. Tesi di Dottorato di Ricerca, Carnegie Mellon Univ. Tech. Report CMU-CS-86-123, Dept. of Computer Science, Carnegie Mellon Univ., Pittsburgh, Pennsylvania, U.S.A.

Rutten J. J. M. M. (1996). Elements of generalized ultrametric domain theory. *Theoretical Computer Science*, **170**(1-2), 349–381.

Sambin G. (1987). Intuitionistic formal spaces: a first communication In *Mathematical Logic and its Applications*. A cura di Skordev D., pp. 187–204. Plenum Press, New York.

Sambin G. (2003). Some points in formal topology. *Theoretical Computer Science*, **305**(1-3), 347–408.

Sangiorgi D. (2009). On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems*, **31**(4).

Sangiorgi D. (2012). *Introduction to Bisimulation and Coinduction*. Cambridge University Press.

Scott D. S. (1971). The lattice of flow diagrams In *Symposium on Semantics of Algorithmic Languages*. A cura di Engeler E., volume 188 di *Lecture Notes in Mathematics*, pp. 311–366, Berlin. Springer-Verlag.

Selinger P. (2011). A survey of graphical languages for monoidal categories In *New Structures for Physics*. A cura di Coecke B., volume 813 di *Lecture Notes in Physics*, pp. 289–355. Springer-Verlag.

Stark E. (1989). Connections between a concrete and an abstract model of concurrent systems. In *Mathematical Foundations of Programming Semantics* in Lecture Notes in Computer Science, numero 442, pp. 53–79. Springer-Verlag.

Terui K. (2011). Computational ludics. *Theoretical Computer Science*, **412**(20), 2048–2071.

Turi D.; Rutten J. J. M. M. (1998). On the foundations of final coalgebra semantics. *Mathematical Structures in Computer Science*, **8**(5), 481–540.

Uustalu T.; Vene V. (1999). Primitive (co)recursion and course-of-value (co)iteration, categorically. *Informatica, Lithuanian Academy of Sciences*, **10**(1), 5–26.

van Glabbeek R. J. (2006). On the expressiveness of higher dimensional automata. *Theoretical Computer Science*, **356**(3), 265–290.

Wadler P. (2014). Propositions as sessions. *Journal of Functional Programming*, **24**, 384–418.