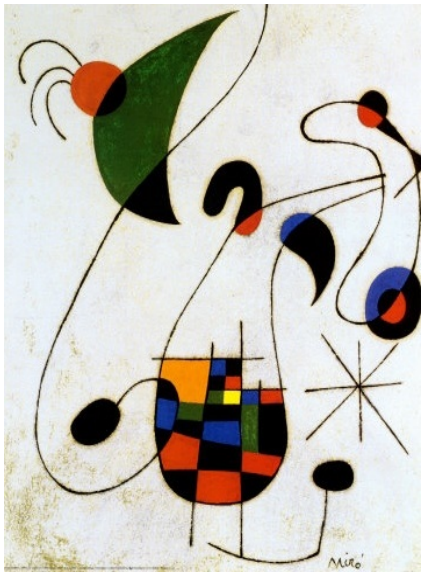
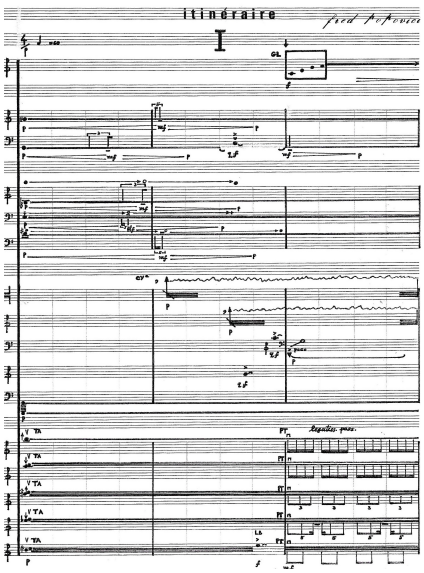
Fred Popovici *Gheorghe M. Ştefan*

Sunet şi complexitate

*

Strategii nelineare în compoziţia muzicală





Editura XXX
2013

Proiect cofinanţat din Fondul Social European prin Programul Operaţional Sectorial pentru Dezvoltarea Resurselor Umane 2007-2013
Investeşte în oameni!

Programul Operaţional Sectorial pentru Dezvoltarea Resurselor Umane 2007-2013.
Axa prioritară 1. Educaţia şi formarea profesională în sprijinul creşterii economice şi dezvoltarii bazate pe cunoaştere.
Domeniul major de intervenţie 1.2. Calitate în învăţământul superior.
Proiect: Sistem integrat de programare de masterat în domeniul ingineriei de sunet, imagine şi al aplicatiilor multimedia.
Contract: POSDRU/86/1.2/S/61810

.

Motto:
Ce se întâmplă cu legile fizicii într-un lift care se prăbuşeşte?

A. Einstein

Introducere

Această carte dezvoltă un proiect de cercetare teoretică şi desfăşurare practică, a strategiei decizionale din compoziţia muzicală, ce emană dintr-un demers filosofic recent. Sfera abordată este cea a investigaţiilor sunetului, a transformărilor lui in timp real şi a elaborării organismelor sonice şi compoziţiei de tip complex.

Cu alte cuvinte, prezentul demers derivă, în mod coerent, dintr-un alt demers, paralel, din spaţiul lingvistic/conceptual. *De-construcţia* este un concept – sau un anticoncept – forjat de filosoful francez Jacques Derrida, în scopul ambiguizării semantice a operelor standardizate din istoria culturii şi pentru a le determina conlucrarea la noi proiecţii semantizate, dez-ideologiza(n)te. În lucrările faimoase, deja, *De la grammatologie* [19], *La dissémination* [21], *L'écriture et la différence* [20], Derrida ne descoperă o dinamică precisă a distorsiunilor şi re-facerilor scriiturii.

După Derrida, de-construcţia urmează două faze: *deplasarea* şi *răsturnarea*. "Subiectul scriiturii este un sistem de relaţii între structuri" [20]: deci, ceva ce nu ar trebui să mai aibe o formă aptă a induce certitudini [21]. Se erodează, astfel, imaginea unui principiu ultim, "totalizant", ce opreşte înlănţuirea continuă a scriiturii [19].

Răsturnarea – ludică *par excellence* – ce confirmă sistemul de referinţă, de de-construcţie, exercitând libertatea de distribuire a elementelor ...

Deplasarea – consecvenţă percepută din exterior a răsturnării – un indecidabil manifest în succesiunea temporală a unor poziţii statice ...

Un simplu exemplu de "orientare" muzicală, chiar dacă nu riguros congruent cu ce urmează, poate, întrucitva, să propună o clarificare. Beethoven a compus în sistemul sonor temperat (generat de puterile lui $\sqrt{2}$) şi după un principiu organizatoric al formei teleologic (finalist). Trecerea paradigmei frecvenţiale în alt sistem (spre exemplu: scara pentatonică chineză) poate fi echivalată unei *deplasări*; iar convertirea strategiei formei într-una constant

repetitivă (exemple aproximative: melodia infinită" la Wagner[1], Gurdjieff interpretat de Keith Jarrett[2], sau minimalismul lui Phillip Glass[3]) poate fi echivalată unei *răsturnări*. De-construcţia se vrea a fi avansarea – niciodată oprită de limitele discursului – în însuşi prezentul unui act in-formativ a franjurilor unui (alt) act, consumat, instituit drept canon în trecut (oricare ar fi acesta din urmă).

Textul pe care îl propunem pe tot cuprinsul cărţii se constituie într-un program concret şi imediat convertibil în *praxis*. Programul are trei paliere de organizare:

palierul conceptual : prin investigarea teoretică a strategiilor de organizare a spaţiului sonor;

palierul interactiv : prin dezvoltarea concretă a tehnologilor de execuţie;

palierul aplicativ : prin desfăşurarea concretă a programului, în spaţiul instituţionalizat şi prin propunera unui nou context arhitectural de elaborare şi execuţie.

Palierul conceptual

Travaliu gramatical

Începutul demersului îl constituie lucrul cu gramaticile generativ-transformaţionale (de tip 2, independente de context), prin aplicaţii asupra materialelor din istoria muzicii. Dacă admitem ideea că fiecare din noi este un "procesor" de limbaj, atunci această realitate are două ipostaze: "recunoscător" al unor secvenţe ca aparţinând unui limbaj şi "generator" al unor secvenţe ce aparţin unui limbaj.

Prin *analiză*, se prelevează paradigma textului ales (se stabileşte alfabetul, apoi vocabularul, V) si se identifică regulile de sintagmatizare a acesteia (G) şi se stabileşte simbolul de start S. Prin *sinteză*, se propune (injectează) un vocabular nou (V'), un set asociat de reguli (G') şi simbolul de start, S'. Se produce extensia transformaţională $G \to G'$, ce va permite traducerea din L în L'.

Schematic, avem următoarele corespondenţe:

[1] `http://www.youtube.com/watch?v=vpQD6_Lyp7M`
[2] `http://www.youtube.com/watch?v=Bb0kbL1YTbk`
[3] `http://www.youtube.com/watch?v=il4VDf-ugPI`

$$\begin{gathered}\text{analiza texului clasic} \rightarrow \text{sinteza noului text} \\ \text{vocabular } V \rightarrow \text{vocabular } V' \\ \text{set reguli } G \rightarrow \text{set reguli } G' \\ \text{limbajul sursă } L(V,G,S) \rightarrow \text{limbajul destinaţie } L'(V',G',S')\end{gathered}$$

Analiza: pornind de la S în L se stabileste secvenţa de reguli (producţii) din G - $(p_1, p_2, , ...p_n)$ – prin care se poate genera un element ("organism") aparţinând lui L.

Sinteza: pornind de la S' în L', se aplică secvenţa de reguli (producţii) corespondente din G' - $(p'_1, p'_2, , ...p'_n)$ – prin care se poate genera un organism" aparţinand lui L'.

Concret, pentru lucrul efectiv cu calculatorul, este nevoie de utilizarea unor rutine în vederea însuşirii lucrului cu gramatici genenerative, prin stabilirea:

- mecanismelor specifice de definire a alfabetului (vocabularului) asociat unui limbaj muzical;
- mecanismelor specifice de definire a regulilor de generare;
- mecanismelor specifice de definire a corespondenţei $G \rightarrow G'$ (un prim pas în care contribuţia creatoare a "beneficiarului", în consecinţă, compozitorul, poate interveni)

Deoarece corespondenţa dintre G si G' nu este strict determinstă (unei producţii in G îi pot corespunde mai multe in G'), traducerea poate asocia unui organism sursă mai multe organisme destinaţie. Procesul de selectie a "organismelor" obţinute, primul pas către producţia unei compoziţii, poate fi realizat prin diferite moduri de estimare a coerenţei demersului efectuat. Preselecţia formală este urmată de o selecţie personală, prin care actul compoziţional beneficiază de contribuţia creatoare a compozitorului (un al doilea moment de intervenţie umană).

Travaliu fractal

Răspândirea fractalilor în spaţiul muzical determină acest program să examineze un mod aparte şi, oricum, coerent prin care se disociază de alte demersuri:

- se propune un nou concept de utilizare a lor, conform unei strategii decizionale personale;

- se propune flexibilizarea formei muzicale eliberată de procedee teleologice (simetrii previzibile, acumulări, descrieri figurative etc.). Se urmăreşte, spre exemplu, disiparea formei până la "zgomot" informaţional, urmată de reinjecţia ordinii până la nivelul la care se (re)echilibrează ordinea cu "dezordinea", astfel încât să se creeze premisele emergenţei unei "forme informale".

Un concept ce este liantul demersului – autosimilaritatea – face ca fractalii, în care se transformă "organismele" de la I, să devină o varietate fenomenologică expresivă prin identitatea procedeelor formatante.

Schemă posibilă:

"organisme" generative →
fractali (regulaţi/aleatorii) →
forme cu grad ridicat de concepere complexă neteleologică.

Procesul este similar celui studiat de D. Hoffstaedter în *Metamagical Themas* [23] pentru a ilustra fenomenul creaţiei muzicale.

Fractalii aplicaţi sistemic asupra "organismelor" rezultate din A.I. reprezintă un proces atât de *deplasare*, cât şi de *răsturnare* din *de-construcţia* anvizajată, având drept rezultat un organism fractal.

Din considerente de claritate sporită, vom exemplifica vizual procesul sonor avut în vedere. Folosim pentru acesta reprezentarea grafică a mulţimii lui Cantor numită "*middle-third-erasing*". Mulţimea menţionată, dată iniţial prin mulţimea punctelor unui segment de dreaptă, se poate dezvolta prin procesul iterativ ilustrat în Figura 1:

Figura 1: **Mulţimea lui Cantor, generată iterativ prin suprimarea treimii mediane.**

Din segmentul iniţial a fost înlaturată, în primul pas, treimea mijlocie. Au rezultat două segmente cărora li se aplică aceeaşi operaţie, ş.a.m.d. În urma a 4 aplicaţii, rezultă 16 segmente, fiecare cu o lungime de 1/81 din lungimea iniţială a segmentului.

Pentru a exemplifica generarea unui organism fractal (vizual), vom considera un segment, notat $org0$, ca fiind organisml generat prin transformarea gramaticală. Un proces de constructie fractală în 3 paşi e ilustrat în Figura 2.

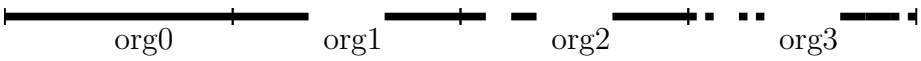


Figura 2: **Fractalizarea organismului** $org0$**.**

În primul pas, organismului $org0$ i se adaugă $org1$ obţinut prin aplicare asupra întregii secvenţe $org0$ a regulii fractale anterior enunţate. În pasul doi, se adaugă $org2$, obţinut din $org1$, căruia i s-a aplicat fractalizarea astfel încât ultimele 2 treimi să rămână neafectate. În ultimul pas se adauga $org3$, construit astfel încât numai primele 5 şesimi din partea neafectata în pasul anterior rămân în continuare neafectate de aplicarea fractalizării. La sfârşitul acestui proces, numai o fracţiune egală cu

$$1 - (1/3 + 1/9 + 3/27) = 12/27$$

din organismul iniţial a mai rămas neafectată de procesul de fractalizare. Acesta poate continua prin menţinerea unei fracţiuni din ce în ce mai mici neafectate de respectivuul proces. Procesul este stopat prin decizia utilizatorului, atunci când acesta consideră transformarea "suficient" de avansată.

Pentru o regulă fractală dată, procesul este caracterizat de secvenţa de proporţii în care partea netransformată se conservă. În exemplul anterior: $\{2/3, 5/6\}$. Această secventă poate fi una stabilită de utilizator în mod arbitrar ("inspirat") sau poate fi o secventă de probabilitaţi generată de un lanţ markovian asociat unui proces real ales de utilizator.

Travaliu de transformări

Cercetările lui Ilya Prigogine asupra dinamicii evenimenţiale pe axa timpului pot constitui un punct de pornire pentru revizuirea strategiei decizionale privind acel proces temporal numit *formă muzicală*. Aici intervin şi sugestii

acumulate de-a lungul istoriei muzicii (conceptul de formă evolutivă la diverşi compozitori).

Întrucât procese de felul celor considerate aici balansează, permanent, între alcătuiri ce sugerează, la limită, haosul si aplatizări informaţionale vecine cu simplismul, programul de faţă propune un proces de generare a *formei* muzicale bazat pe mecanisme de *transformare* de două tipuri: *transformări discrete* – realizat prin opţiuni revizuite într-un proces bazat pe gramatici generative – şi *transformări continue* – bazat pe bucla de reinjecţie (*the reinjection loop*).

Fenomenul de transformare, discretă sau continuă, se aplică recurent unui organism sonor – anterior supus unui travaliu gramatical şi/sau fractal – astfel încât ponderea componentelor din organismul sonor este modificată de fiecare dată. Aplicarea recurentă este realizată până când utilizatorul consideră că a obţinut o modificare suficient de avansată (un alt moment al intervenţiei decizionale a utilizatorului). Să numim rezultatul acestei operaţii ***organism sonor deformat***. Operaţia îşi găseste un corespondent aproximativ în mecanismul de obţinere variaţiunilor din compozitia convenţională.

Forma muzicală emerge prin compunerea organismelor sonore obţinute recurent ca urmare a unor transformări discrete şi/sau continue. Compunerea se poate realiza în mai multe moduri: serial, paralel sau serie-paralel. Astfel, organismele sonore sunt distribuite succesiv, simultan sau în succesiuni de execuţii simultane.

Aplicarea transformărilor şi compunerii acestora în vederea creşterii controlate a unei structuri poate conduce, dacă secvenţa intervenţiilor utilizatorului în întregul proces transformaţional se dovedeste "inspirată", la emergenţa unei forme muzicale *non-teleologice*.

Procesele de de-construcţie – deplasarea si răsturnarea – sunt implicit antrenate în cele prezentate anterior. Astfel, principiul marii forme "clasice" (în accepţia cea mai generică), dar şi structuralismul secolului al XX-lea, ca şi repetitivismul (minimalismul) etc. sunt evacuate în favoarea *purei deveniri sonore* [22]. Există o legătură ce trebuie sistemic controlată între organizarea generativ-transformaţională, fractali şi ordonarea temporală bazată pe procese non-lineare şi de natură "catastrofică" (René Thom [41]) ce constituie materia acestui travaliu.

Problema expusă – cea a palierul concepual – ar putea fi sintetizată prin aplicarea succesivă a:

paradigmei derivativ-transformaţionale (*deplasare*) →

fractalizării (*deplasare* şi *rasturnare*) →

generării formei prin compunerea transformărilor (*forma sonoră*).

Se generează, astfel, prima formă a partiturii unei compoziţii muzicale, să o numim ***pre-partitură***. Pre-partitura este produsul palierului conceptual al programului. Ea va fi supusă ***palierului interactiv*** în vederea obţinerii ansamblului

compoziţie = *pre-partitură + program de procesare în timp real*

necesar actului interpretativ. Programul de procesare în timp real face obiectul secţiunii următoare.

Palierul interactiv

Paradigme ale sunetului

Proiectul de faţă are, ca al doilea palier de realizare, însuşirea şi dezvoltarea tehnicilor de prelucrare în timp real (*live electronics*) a sunetului, în vederea valorificării, extinderii şi concretizării (manipulării) eficiente a pre-partiturii obţinute în palierul conceptual. Interpretarea pre-partiturii se realizează de către instrumentişti, cu instrumente convenţionale, al căror sunet este preluat direct şi prelucrat în *timp real* pentru a se obţine materializarea efectivă a compoziţiei. Execuţia muzicală este astfel asistată în timp real de tehnicile palierului interactiv, care vor fi restricţionate la a introduce o întârziere de cel mult 0,2 secunde. De aici, vor rezulta performanţele computaţionale ale dispozitivelor implicate în execuţia muzicală. Prelucrarea în timp real a semnalelor produse direct de interpreţii umani se va face conform unui program de procesare în timp real, rulat de un sistem de procesare digitală a semnalului audio. Rularea programului de procesare în timp real este realizată sincronizat de interpretarea pre-partiturii de către instrumentişti. Acest program de procesare în timp real este parte a compoziţiei generată tot de către compozitor.

Pentru conceperea şi rularea programului de procesare în timp real sunt folosite instrumente software comerciale (MAX/MSP, PRO TOOLS, AUDIO SCULPT, LAB VIEW, PURE DATA).

Aceste tehnici au caracteristici definite, stocate în programe deja existente sau care pot fi generate în continuare, prin varii metode. Utilizarea lor are ca scop extensia instrumentală (cu tot ceea ce implică aceasta) ce poate merge până la definirea de *noi instrumente*.

Paradigme tehnice

Tehnicile de prelucrare în timp real a sunetului pot fi divizate (după criterii de eventuală utilizare) în:

- tehnici *paradigmatice*, care procură material de lucru (spectre de armonice sau de inarmonice, etc.) sub forma unui repertoriu ("dicţionar") de tehnici electroacustice;
- tehnici *sintagmatice* (sunt procedee care simulează procese acustice sau electroacustice) prin care procedeeele paradigmatice sunt convenabil inlănţuite.

Folosim cuvintele *paradigmă* şi *paradigmatic* în accepţiunea lor cea mai uzuală, aceea de listă cuprinzând cuvinte stocate după un anumit procedeu. Exemplul la îndemână este, fireşte, dicţionarul. Ele se structurează după anumite reguli şi conţin, în genere, materialul tuturor discursurilor de natură lingvistică. De observat, aici, că o paradigmă este independentă de ordinea temporală.

Sintagmă şi *sintagmetic* se referă, dimpotrivă, la ordini temporale; aceste noţiuni presupun o anumită aşezare funcţională a termenilor alcătuitori. Când se spune, de pildă, că "tactica politică" este o sintagmă, se presupune că există o legătură – de obicei, biunivocă – între cei doi temeni, pe lângă obsevarea unui liant de tip gramatical: tactica este, aici, un substantiv, iar politică este un adjectiv.

O idee importantă este aceeea a *sintagmatizării* unei liste paradigmatice. Ideea se referă la faptul că componentele unei liste – în cazul muzicii, există mai multe liste: a sunetelor, a valorilor temporale ale acestora, etc. – sunt alese pentru a provoca percepţia unei constelaţii numită *structură*. Adică, o reţea de relaţii (relaţii de relaţii) ce implică raporturi reciproce, dar mai ales, intenţia determinării unei semnificaţii.

Înlanţuirea tehnicilor paradigmatice prin tehnici sintagmatice se materializează într-un program care rulează *trigger*-at de execuţia intrumentală a pre-partiturii. Performanţa publică este astfel organizată încât în sală este

receptionat preponderent *output*-ul programului de procesare în timp real. În ultimul capitol al lucrării este propusă arhitectura FREGE, arhitectura unui sistem care detectează automat semnalele de *trigger*-are folosite pentru a sincroniza generarea efectelor *live electronics*.

Paradigme de concordanţă

Problema esenţială ce apare este una metodologică: setul (seturile) de procedee din *live electronics* trebuie să fie acordate coerent (prin selecţie realizată de compozitor) cu coordonatele travaliilor anterioare ce au generat pre-partitura. Este realizată concordanţa dintre procedeele aplicate în prima etapă, de construcţie a pre-partiturii, şi procedeele pe care se bazează conceperea programului de procesare în timp real.

Sensul pe care-l are căutarea unor paradigme de concordanţă poate fi explicat, aproximativ, printr-o analogie cu compoziţia clasică. Am putea considera pre-partitura ca o scriitură unde instrumentele implicate în execuţia unei compozitii clasice sunt aproximativ sau deloc specificate. Programul de procesare în timp real ar putea fi interpretat ca procedeul prin care sunt specificate în pre-partitură instrumentele. Avantajul suplimentar al demersului propus este acela că "instrumentele" instanţiate prin programul de procesare în timp real pot avea caracteristici care evoluează în timpul execuţiei, sau pot fi schimbate de la o execuţie la alta în funcţie de intenţiile compozitorului sau interpreţilor, care se pot astfel adapta unor evoluţii imprevizibile sau unor spaţii culturale diferite.

Modalităţile de concordare între strategia decizională, ce are ca rezultat obiectul sonor definit în secţinea **Palierul conceptual**, sub forma prepartiturii, şi tehnologiile definite în secţiunea **Palierul interactiv** reprezintă un obiectiv fundamental al proiectului. Aceasta înseamnă asimilarea unei tehnologii coerente cu instrumentul propriu-zis de organizare a spaţiului unui demers conceptual. Avem de a face cu un proces de învăţare, ca parte esenţială a desfăşurării proiectului. Paradigmele de concordanţă vor emerge din rularea proiectului prin succesive aplicaţii reale ce presupun **Palierul conceptual**, **Palierul interactiv** şi execuţia publică a *compoziţiei = prepartitura+program de procesare in timp real*.

Rularea acestui proiect se poate institui într-un proces cultural iterativ care va fi capabil să impună noi paradigme de concordanţă, consistente cu fenomenologia organizării şi perceperii spaţiului sonor.

Palierul aplicativ

Al treilea palier al proiectului prezentat aici se referă la organizarea sa concretă, la dimensiunea lui aplicativă. Este, în acelaşi timp, un proiect formativ/didactic, de cercetare/dezvoltare/producţie şi diseminare/interacţie.

Aspectul formativ/didactic

Proiectul se adresează unei constelaţii nuanţat conturată de potenţiali beneficiari. Astfel, acesta poate fi accesat de studenţi în cadrul programelor de licenţă, masterat sau doctorat. În acest sens, lucrarea de faţă este, desigur, un *text book* de folosit în practica învăţământului.

Cursurile şi activităţile practice sunt legate de tehnicile convenţionale de procesare a semnalului sonor, dar şi de tehnicile originale dezvoltate în cadrul proiectului. Într-o fază superioară de însuşire a acestora se poate trece la elaborarea de proiecte de cercetare avansată şi de proiecte compoziţionale.

Aspectul de cercetare/dezvoltare/producţie

Se adresează celor care posedă nivelul necesar pentru angajarea în proiecte personale şi au nevoie de un cadru pentru desfăşurarea strategiei propuse de aceştia. Proiectul poate susţine, logistic şi tehnic, acest tip de demersuri.

Aspectul de diseminare/interacţie

Trebuie arătat faptul că proiectul prezent, descris in linii generale, este unic în Europa de Est. El poate duce la colaborări şi travalii îmbinate cu alte centre din lume. Formele de colaborare avute în vedere sunt:

- workshop-uri;
- diverse forme de mixare a travaliului cu alte centre internaţionale.

Concluzii

Care este menirea proiectului pe care-l prezentăm, detaliat, în componentele sale, în cele ce urmează?

- În primul rând , o reconsiderare, graţie de-construcţiei, a potenţialului – pentru creaţia nouă, prospectivă – existent în spaţiile artei clasice, ratificată de istorie, potenţial încă neexplorat.

- În al doilea rând, o modalitate *sui generis* (în măsura în care acest lucru este, încă, posibil?) de îmbinare a unor tehnici avansate, care şi-au demonstrat, fiecare, avantajele în câte un domeniu de cercetare într-o sinteză ce ambiţionează eficienţa de astăzi, ce ar putea-o egala pe cea dovedită de alte strategii decizionale, în alte perioade ale culturii muzicale.

- În al treilea rând, uneltele cele mai avansate ce există astăzi pentru tratamentul şi prelucrarea sunetului şi semnalului sonor (atât la nivel *hard* cât şi *soft*) sunt anvizajate prin prisma unui program de lucru coerent, ce asamblează viziunea creatoare şi un set de unelte menite a le concretiza.

Proiectul înfăţişat în textul prezent are ca ţintă elucidarea unor probleme, a unor dileme pe care, fără îndoială, şi le pun deopotrivă compozitori, muzicologi, interpreţi ai muzicii contemporane, dar şi cercetători din sfera investigaţiilor electroacustice şi ale informaţicii muzicale.

Septembrie, 2012 Fred Popovici

Cuprins

Partea I

Procesarea componistică neliniară

Capitolul 1

Limbaje şi gramatici formale

Studiul limbajelor formale din perspectiva gramaticilor generative este declanşat în deceniul 6 al secolului al XX-lea de către Noam Chomsky. *Gramatica de la Porte Royal* publicată în 1660 de către Antoine Arnauld şi Claude Lancelot a fost o tentativă temerară care, prin nereuşita ei, a arătat că realitatea limbajului natural nu poate fi surprinsă integral într-un formalism. Poate fi formalizat riguros, printr-un număr finit de reguli, numai un proces care a fost conceput artificial prin respectarea unor restricţii. În primul deceniu al producţiei de serie a calculatoarelor electronice, entuziasmul unui început promiţător face ca principalii actori să iniţieze proiecte foarte ambiţioase ce se vor dovedi ulterior parţial utopice. Inteligenţa artificială, sistemele autoreproductibile, traducerea automată sunt numai câteva proiecte care au determinat progrese uluitoare în ştiinţa calculului automat, chiar dacă nu şi-au îndeplinit toate promisiunile. Unul dintre acestea a fost cel iniţiat de Noam Chomsky.

Lingvistica computaţională, teoria compilatoarelor, tehnicile de securizare a sistemelor de calcul, teoria generală a computaţiei beneficiază de rezultatele teoretice obţinute în mai bine de o jumătate de secol de cercetare iniţiată de cercetările lingvistice ale lui Noam Chomsky. În orice domeniu unde se pot evidenţia limbaje formalizabile sau măcar parţial formalizabile, teoria limbajelor formale îşi poate dovedi utilitatea.

Rezultatele teoretice obţinute spun lucruri semnificative şi despre cei ce folosesc limbaje formale mai mult sau mai puţin expresive. Dacă acceptăm ipoteza conform căreia gramaticile generative se află în mare măsură la baza mecanismelor prin care un limbaj se manifestă, atunci se pot face afirmaţii pertinente şi asupra utilizatorilor umani ai limbajelor formale sau parţial

formalizabile. Un limbaj parţial formalizabil este cel muzical.

Începem printr-o serie de definiţii care au rolul de a crea cadrul riguros în care vom aborda problema limbajelor formale.

Definitia 1.1 *Un* **alfabet** *este mulţimea finită a simbolurilor distincte, A, acceptată de un mecanism automat de recunoaştere sau de generare.*

◇

Chiar dacă expresiile formulate într-un limbaj formal pot fi oricât de lungi, simbolurile trebuie să ia valori într-o mulţime finită pentru a putea construi maşini şi tehnici algoritmice finit dimensionate prin care expresiile să poată fi ***recunoscute*** sau ***generate***. Recunoaştera şi generarea sunt cele două procese esenţiale legate de limbajele formale.

Definitia 1.2 *Mulţimea tuturor şirurilor peste un alfabet finit A este notată prin definiţie cu* A^*.

◇

Mulţimea A^* este infinită, pentru că nu este limitată lungimea şirurilor generate. Finite trebuie să fie maşinile şi tehnicile algoritmice asociate.

Exemplul 1.1 *Dacă* $A = \{0, 1\}$, *atunci*

$$A^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, 001, ...\}$$

unde λ *este elementul nul.*

◇

Definitia 1.3 *Un* **limbaj**, *L, definit peste alfabetul A este o submulţime din* A^*; $L \subset A^*$.

◇

Exemplul 1.2 *Dacă* $A = \{a\}$, *atunci* $L = \{a^k \mid 0 < k < n\}$ *este limbajul finit al şirurilor formate din cel mult* $n - 1$ *a-uri.*

◇

Un limbaj poate fi definit prin:

- enumerarea componentelor unei submulţimi din A^*

- specificarea regulilor de generare a unei submulţimi din A^*.

Primul procedeu este incomod. Nu poate fi aplicat decât limbajelor cu un număr finit de elemente. Cel de al doilea procedeu a condus la definirea conceptului de ***gramatică generativă***.

Aplicaţia cea mai importantă a gramaticilor generative este în domeniul trducerii dintr-un limbaj formal într-altul. Spre exemplu, din limbajul de programare C++ în limbajul de asamblare al unui procesor Intel. În această carte, tot despre traducere va fi vorba. De data aceasta, vom porni de la structura simplificată a limbajului asociat unui grup de compoziţii muzicale (spre exemplu, sonatele pentru pian de Mozart) şi vom realiza translatarea într-un limbaj formal construit de un compozitor contemporan.

1.1 Gramatici generative

În această secţiune, vom folosi texte anterior publicate [2] [18] [37] [40] pentru a specifica ceea ce este o gramatică generativă. Ne vom limita la ceea ce este necesar demersului nostru, atât din punct de vedere tehnic, cât şi din punctul de vedere al înţelegerii proceselor componistice investigate.

Unui limbaj formal specificat i se poate asocia o gramatică ce îl descrie în totalitate. Limbajul muzical nu este un limbaj strict formalizabil. În calitatea lui de limbaj ce a rezultat printr-un proces istoric emergent, are multe în comun cu limbajele naturale. Restricţiile tehnice şi stilistice ce l-au marcat în cadrul aceleiaşi evoluţii au indus aspecte formale importante. Putem spune astăzi că limbajul muzical al anumitor epoci, sau al anumitor compozitori, formele muzicale în general prezintă multe aspecte formalizabile. Pe aceste caracteristici formalizabile se bazează demersul nostru.

Definitia 1.4 *O gramatică [12] este qvadruplul*

$$G = (N, T, P, S)$$

unde:

N *: este mulţimea finită a simbolurilor neterminale*

T *: este mulţimea finită a simbolurilor terminale*

P *: este mulţimea finită a producţiilor de forma* $p \to q$*, cu:*

$$p \in \{N \cup T\}^* \setminus \{\lambda\}$$

unde p *conţine cel puţin un element din* N

$$q \in \{N \cup T\}^*$$

S *: este simbolul de start;* $S \in N$*.*

$\diamond$

Simbolurile neterminale N sunt necesare pentru a permite controlul creşterii şi conturării formei secvenţei generate. Oprirea procesului de generare se face atunci când expresia a ajuns să nu mai conţină nici un neterminal. Acesta este motivul pentru care partea stângă a oricărei producţii p conţine cel puţin un neterminal. Gramatica generativă oferă cadrul în care un şir de simboluri se poate forma coerent; nu ne ajută la obţinerea unei *anumite* secvenţe. Nu ne garantează decât faptul că toate secvenţele generate vor aparţine limbajului pe care l-am definit.

Conceperea unei gramatici asociate unui limbaj este o problemă esenţială. Se poate porni de la definiţia formală a limbajului, dacă ea există. O altă soluţie, ce va fi adoptată în demersul nostru, este aceea de a porni de la eşantioane semnificative ale limbajului. Nu există un procedeu algoritmic care să garanteze o soluţie (optimă).

Exemplul 1.3 *Fie gramatica:*

$$G_1 = (\{S, A\}, \{a, b, c\}, \{S \to aAa, A \to aAa \mid bAb \mid c\}, S)$$

Un exemplu de generare este:

$$\begin{gathered} S \to aAa \\ aAa \to aaAaa \\ aaAaa \to aabAbaa \\ aabAbaa \to aabaAabaa \\ aabaAabaa \to aababAbabaa \\ aababAbabaa \to aababcbabaa \end{gathered}$$

Şirurile generate sunt simetrice şi cresc în două puncte din şir. Procesul de generare se opreşte atunci când nu mai poate fi aplicată nici una dintre regulile de generare. Ordinea de aplicare a producţiilor nu este impusă de gramatică, ci exclusiv de "voinţa" ce acţionează în procesul de genereare.

◇

Definitia 1.5 *Deducţia în n paşi lui p_n din p_0 in G este*

$$p_0 \to p_1 \to p_2 \ldots \to p_n = p_0 \overset{n}{\underset{G}{\Rightarrow}} p_n$$

dacă p_i se obţine din p_{i-1} prin aplicarea unei producţii din P, $p_i \in \{N \cup T\}^*$ pentru $i = 0, 1, \ldots n-1$ şi $p_n \in T^*$.

◇

Care este secvenţa de producţii care poate genera un şir dat, pornind din S, va fi o problemă cu care ne vom confrunta în aplicarea concretă a unor gramatici pe care le vom construi.

Definitia 1.6 *Limbajul generat de gramatica $G = (N, T, P, S)$ este:*

$$L(G) = \{p \mid p \in T^*, S \overset{*}{\underset{G}{\Rightarrow}} p\}$$

care reprezintă mulţimea tuturor şirurilor generate, într-un număr oarecare de paşi, folosind gramatica G pornind de la simbolul de start S.

◇

Construcţia unei gramatici depinde de modul în care este specificat limbajul. Cel mai simplu este cazul în care avem un limbaj specificat formal într-un mod riguros. Exemplul cel mai bun este cel al unui limbaj de programare. Pentru demersul nostru componistic, soluţia va fi mai dificilă pentru că nu putem folosi decât eşantioane considerate semnificative.

Un exemplu mai puţin artificial decât cel anterior este cel ce urmează, care va exemplifica felul în care se poate controla, într-un caz simplificat, buna formare a expresiilor algebrice.

Exemplul 1.4 *Gramatica G_2 este folosită pentru a genera expresii algebrice corecte.*

$$G_2 = (\{S, M, F\}, \{a, +, *, (,)\}, P, S)$$

unde:

$$P = \{S \to S + M \mid M, M \to M * F \mid F, F \to a \mid (S)\ \}$$

Să considerăm expresia:

$$a + a * (a + a)$$

Gramatica G_2 o generează după cum urmează:

$S \to S + M$;
$S + M \to M + M$; *se aplică producţia* $S \to M$
$M + M \to F + M$; *se aplică producţia* $M \to F$
$F + M \to a + M$; *se aplică producţia* $F \to a$
$a + M \to a + M * F$; *se aplică producţia* $M \to M * F$
$a + M * F \to a + F * F$; *se aplică producţia* $M \to F$
$a + F * F \to a + a * F$; *se aplică producţia* $F \to a$
$a + a * F \to a + a * (S)$; *se aplică producţia* $F \to (S)$
$a + a * (S) \to a + a * (S + M)$; *se aplică producţia* $S \to S + M$
$a + a * (S + M) \to a + a * (M + M)$; *se aplică producţia* $S \to M$
$a + a * (M + M) \to a + a * (F + M)$; *se aplică producţia* $M \to F$
$a + a * (F + M) \to a + a * (a + M)$; *se aplică producţia* $F \to a$
$a + a * (a + M) \to a + a * (a + F)$; *se aplică producţia* $M \to F$
$a + a * (a + F) \to a + a * (a + a)$; *se aplică producţia* $F \to a$

expresia

$$a + (a * +$$

spre exemplu, nu poate fi generată folosind G_2, deoarece algoritmul de identificare a şirului de producţii care o generează în G_2 va rejecta expresia ca fiind ne-generabilă.

◇

Exemplul 1.5 *Fie gramatica:*

$$G_3 = \{\{S, B\}, \{a, b\}, \{S \to aS \mid aB, B \to bB \mid b\}, S\}$$

O posibilă generare este:

$S \to aS$
$aS \to aaS$
$aaS \to aaaB$
$aaaB \to aaabB$

$aaabB \to aaabbB$
$aaabbB \to aaabbbB$
$aaabbbB \to aaabbbb$

Limbajul generat este:

$$L(G_3) = \{a^n b^m \mid n, m \geq 1\}$$

$L(G_3)$ generează un şir de n a-uri urmat de un şir m de b-uri.
$\diamond$

Gramatica din exemplul anterior generează şiruri care cresc la un singur capăt.

Exemplul 1.6 *Limbajul*

$$L(G_4) = \{a^n b^n \mid n \geq 1\}$$

poate fi generat de gramatica

$$G_4 = \{\{S\}, \{a, b\}, \{S \to aSb \mid ab\}, S\}$$

$L(G_4)$ generează un şir de n a-uri urmat de un şir ce conţine tot atâtea b-uri.
$\diamond$

Limbajul generat de gramatica G_4, $L(G_4)$, este mai "expresiv" decât limbajul generat de gramatica G_3, $L(G_3)$, deoarece ambele generează a-uri urmate de b-uri, dar G_4 satisface o condiiţie suplimentară: a-urile şi b-urile sunt la fel de multe.

Exemplul 1.7

$$G_5 = \{\{S, B\}, \{a, b, c\}, P, S\}$$

unde mulţime regulilor de producţie, P, are următorul conţinut:

$S \to aBSc \mid abc;$
$Ba \to aB;$
$Bb \to bb;$

O posibilă generaare se derulează după cum urmează:

$S \to aBSc$
$aBSc \to aBaBScc$
$aBaBScc \to aBaBabccc$
$aBaBabccc \to aaBBabccc$
$aaBBabccc \to aaBaBbccc$
$aaBaBbccc \to aaaBBbccc$
$aaaBBbccc \to aaaBbbccc$
$aaaBbbccc \to aaabbbccc = a^3b^3c^3$

Este evident că:

$$L(G_5) = \{a^n b^n c^n \mid n \geq 1\}$$

$\diamond$

Producţiile din G_5 sunt dependente de context, deoarece, spre exemplu, ultima producţie substituie pe B cu b dar numai în contextul în care B precede un b.

Exemplul 1.8 *Fie gramatica:*

$$G_6 = \{\{S, A, B, C, D\}, \{a, b\}, P, S\}$$

unde P conţine:

$S \to CD$
$C \to aCA \mid bCB$
$AD \to aD$
$BD \to bD$
$Aa \to aA$
$Ab \to bA$
$Ba \to aB$
$Bb \to bB$
$C \to \lambda$
$D \to \lambda$

Reamintim că λ este elementul nul. Ultimele două reguli permit dispariţia unor elemente din şir pe parcursul procesului de generare. O derivare posibilă pornind din S este:

$S \to CD;$	
$CD \to aCAD;$	*s-a aplicat* $C \to aCA$
$aCAD \to abCBAD;$	*s-a aplicat* $C \to bCB$
$abCBAD \to abBAD;$	*s-a aplicat* $C \to \lambda$
$abBAD \to abBaD;$	*s-a aplicat* $AD \to aD$
$abBaD \to abaBD;$	*s-a aplicat* $Ba \to aB$
$abaBD \to ababD;$	*s-a aplicat* $BD \to bD$
$ababD \to abab;$	*s-a aplicat* $D \to \lambda$

În procesul de generare, şirul nu a crescut consecvent. În anumite etape a avut o lungime mai mare decât şirul final generat.

◇

Primele două reguli de producţie permit şirului generat să crească. Urmează regulile care îl reconfigurează prin producţii dependente de context. Ultimele două producţii permit şirului să-şi reducă dimensiunea prin substituţii cu elementul nul λ.

1.2 Ierarhia gramaticilor generative

În [12] [13] [15] Noam Chomsky a introdus conceptul de ***ierarhie a gramaticilor*** în funcţie de restricţiile impuse regulilor de generare.

Definitia 1.7 *O gramatică generativă G poate fi:*

regulată *sau de **tip 3** dacă fiecare producţie din P este de forma*

$$A \to xB \mid x$$

unde $A, B \in N$ şi $x \in T^*$

independentă de context *sau de **tip 2** dacă fiecare producţie din P este de forma*

$$A \to \alpha$$

unde $A \in N$ şi $\alpha \in (N \cup T)^*$

dependentă de context *sau de **tip 1** dacă fiecare producţie din P este de forma*

$$\alpha \to \beta$$

unde $\alpha, \beta \in (N \cup T)^*$ şi $|\alpha| \leq |\beta|$

fără restricţii *sau de* ***tip 0*** *dacă fiecare producţie din P nu este supusă niciunei restricţii.*

unde cu $|I|$ s-a notat lungimea şirului I.

◇

Între gramaticile de tip 1 şi cele de tip 0 se află, cu certitudine, şi alte gramatici care sunt bazate pe reguli supuse la restricţii mai "slabe" decât cel impuse gramaticilor de tip 1. Interesul pentru ele nu există pentru că limbajele formal cele mai utile s-au dovedit a fi cele de tip 2.

Pentru a evidenţia ierarhizarea printr-o ierarhie de restricţii, vom defini următoarele restricţii pentru producţiile $p \to q$:

R1 : $|p| \leq |q|$

R2 : $|p| = 1$, cu implicaţia evidentă $p \in N$

R3 : $q = \alpha A$, unde $\alpha \in T^*$, $A \in N \cup \{\lambda\}$

Definitia 1.8 *O gramatică generativă G este:*

fără restricţii *sau de tip 0*

dependentă de context *sau de tip 1 dacă satisface restricţia R1*

independentă de context *sau de tip 2 dacă satisface restricţiile R1 şi R2*

regulată *sau de tip 3 dacă satisface restricţiile R1, R2 şi R3*

◇

Notăm cu $\mathcal{L}_i$ mulţimea tuturor limbajelor de tip i, pentru $i \in \{0, 1, 2, 3\}$.

Exemplul 1.9 *În exemplele de gramatici date anterior identificăm următoarele tipuri de gramatici:*

- *$G_3 \in \mathcal{L}_3$ deoarece*
  $G_3 = \{\{S, B\}, \{a, b\}, \{S \to aS \mid aB, B \to bB \mid b\}, S\}$ are numai reguli de producţie de forma: $A \to xB \mid x$

- $G_1, G_2, G_4 \in \mathcal{L}_2$ *deoarece cele trei gramatici*
  $G_1 = (\{S, A\}, \{a, b, c\}, \{S \to aAa, A \to aAa \mid bAb \mid c\}, S)$
  $G_2 = (\{S, M, F\}, \{a, +, *, (\ )\}, P, S)$ *cu*
  $P = \{S \to S + M \mid M, M \to M * F \mid F, F \to a \mid (S)\}$
  $G_4 = \{\{S\}, \{a, b\}, \{S \to aSb \mid ab\}, S\}$
  au producţii de tipul $A \to \alpha$ *unde* $A \in N$ *şi* $\alpha \in (N \cup T)^*$

- $G_5 \in \mathcal{L}_1$ *deoarece*
  $G_5 = \{\{S, B\}, \{a, b, c\}, P, S\}$ *cu*
  $P = \{S \to aBSc \mid abc, Ba \to aB, Bb \to bb\}$ *are producţii de tipul* $\alpha \to \beta$ *unde* $\alpha, \beta \in (N \cup T)^*$ *şi* $|\alpha| \leq |\beta|$

- $G_6 \in \mathcal{L}_0$ *deoarece* $G_6 = \{\{S, A, B, C, D\}, \{a, b\}, P, S\}$ *cu*
  $P = \{S \to CD, C \to aCA \mid bCB, AD \to aD, BD \to bD, Aa \to aA, Ab \to bA, Ba \to aB, Bb \to bB, C \to \lambda, D \to \lambda\}$ *are şi producţii de tipul* $A \to \lambda$ *prin care, în procesul de generare, şirul îşi reduce lungimea.*

◇

Teorema 1.1 $\mathcal{L}_0 \supset \mathcal{L}_1 \supset \mathcal{L}_2 \supset \mathcal{L}_3$.

◇

Demonstraţia se obţine direct prin aplicarea definiţiei 1.8. Consecinţa importantă a teoremei anterioare este aceea că o maşină care recunoaşte orice limbaj din $\mathcal{L}_i$ va recunoaşte şi orice limbaj din $\mathcal{L}_j$ daca $j > i$.

1.3 Traducerea

Traducerea dintr-un limbaj formal sursă, $L(G)$, într-un limbaj formal destinaţie, $L'(G')$, se face utilizând gramaticile generative. Procesul de traducere presupune următoarele etape:

1. se pun în corespondenţă producţiile gramaticilor care definesc limbajele între care se face traducerea

2. se identifică o secvenţă de producţii din G care generează pornind din S elementul $q \in L$ care trebuie tradus

3. se construieşte, conform corespondenţei stabilite la punctul 1, secvenţa de producţii din G' care corespunde secvenţei de producţii identificate la punctul 2

4. se generează, pornind din S', traducerea elementului $q \in L$ în $q' \in L'$ aplicând secvenţa de reguli stabilită la punctul 3.

Exemplul 1.10 *Traducerea unor expresii algebrice* infixate *în expresii* postfixate *se poate face definind, pe lângă limbajul $L(G_2)$ care generează expresii infixate, limbajul $L(G'_2)$ care produce expresii postfixate.*

Limbajul $L(G'_2)$ este definit prin următoarea gramatică:

$$G'_2 = (\{S, M, F\}, \{a, +, *\}, P, S)$$

unde:

$$P = \{S \to SM+ \mid M, M \to MF* \mid F, F \to a \mid S\}$$

Prima etapă este cea în care stabilim corespondenţa între regulile de producţie. Pentru exemplul nostru coresppondenţa este:

reguli sursă	$\longleftrightarrow$	**reguli destinaţie**
(1) $S \to S + M$	$\longleftrightarrow$	$S \to SM+$
(2) $S \to M$	$\longleftrightarrow$	$S \to M$
(3) $M \to M * F$	$\longleftrightarrow$	$M \to MF*$
(4) $M \to F$	$\longleftrightarrow$	$M \to F$
(5) $F \to a$	$\longleftrightarrow$	$F \to a$
(6) $F \to (S)$	$\longleftrightarrow$	$F \to (S)$

În a doua etapă stabilim secvenţa de producţii care produce elementul din L ce se traduce. Pentru expresia:

$$a + a * (a + a)$$

secvenţa de reguli in L a fost (vezi Exemplu 1.4):

$$(1), (2), (4), (5), (3), (4), (5), (6), (1), (2), (4), (5), (4), (5)$$

Rezultă secvenţa de reguli din L' care se va aplica lui S':

$S \to SM+$
$S \to M$
$M \to F$
$F \to a$
$M \to MF*$
$M \to F$

$F \to a$
$F \to S$
$S \to SM+$
$S \to M$
$M \to F$
$F \to a$
$M \to F$
$F \to a$

Ultima etapă a traducerii prespune aplicarea secvenţei identificate lui S', pentru a obţien forma expresiei in limbajul destinaţie, după cum urmează:

$S \to SM+;$	
$SM+ \to MM+;$	*se aplică producţia* $S \to M$
$MM+ \to FM+;$	*se aplică producţia* $M \to F$
$FM+ \to aM+;$	*se aplică producţia* $F \to a$
$aM+ \to aMF*+;$	*se aplică producţia* $M \to MF*$
$aMF*+ \to aFF*+;$	*se aplică producţia* $M \to F$
$aFF*+ \to aaF*+;$	*se aplică producţia* $F \to a$
$aaF*+ \to aaS*+;$	*se aplică producţia* $F \to S$
$aaS*+ \to aaSM+*+;$	*se aplică producţia* $S \to SM+$
$aaSM+*+ \to aaMM+*+;$	*se aplică producţia* $S \to M$
$aaMM+*+ \to aaFM+*+;$	*se aplică producţia* $M \to F$
$aaFM+*+ \to aaaM+*+;$	*se aplică producţia* $F \to a$
$aaaM+*+ \to aaaF+*+;$	*se aplică producţia* $M \to F$
$aaaF+*+ \to aaaa+*+;$	*se aplică producţia* $F \to a$

A rezultat expresia $aaaa+*+$ care este corespondentul postfixat al expresiei infixate $a+a*(a+a)$.

◇

În exemplul dat, fiecărei reguli de producţie din G i-a corespuns o singură regulă în G'. Se pot imagina traduceri în care corespondenţa să nu fie strict univocă. Unei reguli din G i se pot asocia mai mlte reguli în G', iar alegerea aplicării uneia sa alteia poate fi un proces aleator.

1.4 Limbaje şi structuri fizice

Corespondenţa dintre limbajele formale de diferite tipuri şi maşinile asociate este un subiect de mult epuizat. Dar dacă Noam Chomsky a stabilit o ie-

rarhie a limbajelor formale, nu cumva este posibilă şi stabilirea unei ierarhii corespondente în lumea maşinilor asociate? Care este criteriul care le-ar putea ierarhiza? Această secţiune răspunde pozitiv la prima întrebare şi oferă şi criteriul de clasificare a maşinilor.

Pentru sistemele digitale există o ierarhizare [39] care se bazează pe numărul de bucle ce se închid în structurile digitale. Sistemele de ordinul 0 (SO0) – cele combinaţionale – sunt structurate fără nici o buclă internă. Sistemele de ordinul 1 (SO1) – memoriile – sunt caracterizate de o buclă internă. Sistemele de ordinul 2 (SO2) – automatele finite – posedă două nivele de bucle ce se includ. Sistemele de ordinul 3 (SO3) – procesoarele – sunt caracterizate prin închiderea a 3 bucle în structura lor internă. Sistemele de ordinul 4 (SO4)– calculatoarele – au 4 bucle ş.a.m.d. Vom vedea că se va putea evidenţia un paralelism între clasificarea limbajelor formale şi cea a structurilor digitale [38].

În această secţiune vom arăta şi comenta următoarele corespondenţe:

1. limbaje de tip 3 ($\mathcal{L}_3$) $\longleftrightarrow$ maşini cu 2 bucle (SO2)

2. limbaje de tip 2 ($\mathcal{L}_2$) $\longleftrightarrow$ maşini cu 3 bucle (SO3)

3. limbaje de tip 1 ($\mathcal{L}_1$) $\longleftrightarrow$ maşini cu 4 bucle (SO4)

Între-adevăr, după cum vom constata, pe măsură ce "expresivitatea" limbajelor creşte, autonomia maşinilor fizice asociate (sau complexitatea mecanismelor mentale implicate) va trebui să crească şi ea.

Pornim de la următoarea teoremă:

Teorema 1.2 *Limbajele formale generate de gramaticile lui Chomsky pot fi recunoscute/generate optimal de următoarele structuri fizice de circuit:*

1. $\mathcal{L}_3$ *- automate finite*

2. $\mathcal{L}_2$ *- automate cu stivă*

3. $\mathcal{L}_1$ *- automate cu memorie linear marginită*

4. $\mathcal{L}_0$ *- maşini Turing.*

$\diamond$

Teorema este demonstrată în toate manualele care tratează limbajele formale. Ceea ce trebuie să arătăm, în cele ce vor urma, este că tipurile de maşini menţionate sunt caracterizate de numărul de bucle interne sugerat anterior.

Principiul care ne va ghida abordarea în continuare porneşte de la faptul că nu putem accepta decât două tipuri de structuri în conceperea maşinilor fizic realizabile:

- dacă o structură este complexă, atunci dimensiunea va trebui să fie constantă
- dacă o structură are dimensiunea dependentă de n, atunci complexitatea va trebui să fie constantă (trebuie să fie o structură definită recursiv).

Structurile complexe cu dimensiunea dependentă de n sunt, în cazul general, nerealizabile fizic în condiţii tehnologice rezonabile.

1.4.1 Gramatici de tip 3 şi structuri cu 2 bucle (SO2)

Teorema 1.3 *Orice limbaj de tip 3 poate fi recunoscut prin stările finale ale unui semi-automat determinist strict iniţial.*

◇

Într-adevăr, memoria internă a unui semi-automat finit (stare sa internă) este suficientă pentru a recunoaşte şiruri generate de o gramatică reguată, doarece şirul generat de o gramatică regulată creşte la un singur capăt, după reguli independente de context.

Teorema 1.4 *Orice limbaj de tip 3 poate fi generat de un automat finit determinist strict iniţial.*

◇

Din raţiuni similare, un automat finit poate genera orice şir regulat, deoarece generarea, în fiecare etapă, presupune selectarea unei reguli care depinde numai de simbolul generat în etapa anterioară. Starea internă a automautlui finit este suficientă pentru a memora care a fost regula de producţie aplicată în etapa anterioară de generare.

În aceste condiţii, va trebui să răspundem numai la întrebarea: care este numărul minim de bucle care ne permite realizarea fizică unui automat finit.

Teorema 1.5 *Ordinul minim al unui sistem digital care permite construcţia unui automat finit este 2.*

◇

Demonstraţie Ne reamintim că un automat finit este definit prin cvintuplul $A = (X, Y, Q, f, g)$, unde: X este mulţimea finită a intrărilor, Y este mulţimea finită a ieşirilor, Q este mulţimea finită a stărilor interne, $f : X \times Q \to Q$ este funcţia de tranziţie a stării interne iar $g : X \times Q \to Y$ este funcţia de tranziţie a ieşirii. Structura internă a unui automat finit (în versiunea Mealy) este reprezentată în Figura 1.1, unde:

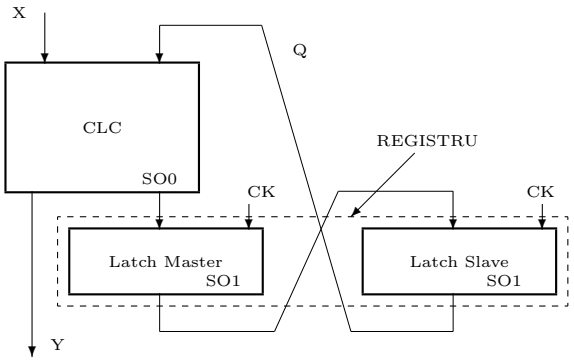


Figura 1.1: Structura unui automat finit tip Mealy

- *CLC* este circuitul logic combinaţional care calculează funcţiile de tranziţie f şi g
- *REGISTRU* este o colecţie de bistabili de tip *master-slave* care funcţionează ca bistabili de tip D cu o organizare internă pe două nivele:
  - *Latch Slave*, care este o colecţie de *latch*-uri destinate stocării stării interne (valoare curentă din Q)
  - *Latch Master*, care este o colecţie de *latch*-uri ce permite închiderea ne-transperentă a buclei în sistem, permiţând şi comportamentul sincron controlabil.

Atsfel, în sistem se impune închiderea a două bucle ce se includ:

- ***prima buclă***, la nivelul fiecărui latch de un bit, ce asigură funcţiile de stocare a stării (în latch-ul master) şi de stocare tranzitorie necesară închidereii transparente, prin latch-ul slave, a celei de a doua bucle

- ***bucla secundă*** – prin CLC, *Latch Master* şi *Slave Latch* – impusă de calculul funcţiei de tranziţie a stării automatului finit f, definită în $X \times \mathbf{Q}$, cu valori în $\mathbf{Q}$. Aici, *Latch Slave* are numai un rol electric, permiţând tranziţia sincronă sub controlul semnalului de ceas al sistemului.

Prima buclă permite, în principal, *memorarea stării* (unde s-a ajuns cu recunoaşterea sau generarea), pe când cea de a doua buclă asigură *comportamnentul autonom* (cum se merge mai departe în procesul de recunoaştere sau generare) al sistemului.

◇

Pe scurt:

- prima buclă permite construcţia circuitului care *stochează*, în latch-ul master, ultimul simbol recepţionat sau generat

- bucla secundă permite *secvenţarea* procesului de recunoaştere sau generare.

Nu este necesară o cantitate de memorie suplimentară, pentru că regulile de producţie sunt simple. Şirul este recunoscut/generat în "timp real" (pe măsură ce este recepţionat/generat) datorită caracterului său regulat. Un şir poate fi rejectat înainte de recepţionarea lui integrală, deoarece apariţia fiecărui simbol este conditionată numai de cel precedent, de care automatul finit poate fi "conştient" datorită regulilor de generare ce sunt numai de tip $A \to xB$. Structuri mai complexe, din categoria SO3, SO4, ... pot fi folosite, dar nu mai puţin autonome, din SO1 sau SO0.

1.4.2 Gramatici de tip 2 şi structuri cu 3 bucle (SO3)

Ne aşteptăm ca trecând la limbajele mai "expresive", de tip 2 (independente de context) să fie necesare maşini cu o autonomie sporită pentru recunoaşterea şi generarea lor. Pot fi automatele folosite la nivelul limbajelor independente de context? Da, dar nu în varianta automatelor *finite*. Numai automatele cu un număr "infinit" de stări vor putea fi utile. Dar dacă vrem să evităm automatele "infinite", atunci va trebui să acceptăm adăugarea unei

bucle suplimentare în sistem. Trecem, astfel, la folosirea unor circuite din clasa SO3.

Dacă pentru recunoaşterea/generarea unui limbaj din $\mathcal{L}_2$ dorim să folosim un automat, atunci numărul de stări ale acestuia va fi $|Q| \in O(n)$, unde n este lungimea maximă a şirurilor de recunoscut/generat, iar $|Q|$ este dimensiunea spaţiului stărilor. Abordarea noastră trebuie să evite o astfel de soluţie şi să se limitezze la folosirea unor structuri simple, cu o complexitate constantă, independentă de n. Dimensiunea (*size*) structurii poate depinde de n, dar nu şi complexitatea.

Să considerăm exemplul, consacrat, al limbajului $\{a^n b^n | n > 0\}$. Dacă s-ar folosi un semi-automat pentru recunoaştere, atunci numărul de a-uri recepţionate trebuie memorat pentru a fi comparat cu cel al b-urilor ce urmează. Singurul loc în care această memorare este posibilă este în spaţiul stărilor, a cărui dimensiune va trebui să devină, astfel, o funcţie de n. Deci, semi-automatul nu va putea fi din categoria celor *finite*, cu dimensiune constantă a spaţiului stărilor. Singura soluţie care permite menţinerea structurii fizice a maşinii în categoria celor simple (cu complexitate constantă, independentă de n) este de a adăuga, pe lângă un automat finit, o resursă simplă cu dimensiunea dependentă de n. Semi-automatul va fi, în consecinţă, înlocuit de structura din Figura 1.2, unde numărătorul reversibil UDCOUNTER (up-down counter) numără prin incrementare a-urile şi se decrementează pentru fiecare b recepţionat. Astfel, automatul finit, FA, ajutat de un numărător rezolvă problema. Sistemul are partea complexă constantă iar partea simplă dependentă de n.

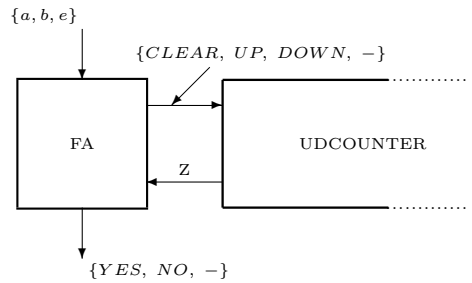


Figura 1.2: Automat finit cu numărător reversibil - un SO3 care poate recunoaşte/genera limbajul $\{a^n b^n | n > 0\}$

Numărătorul este o memorie foarte simplă, care are inconvenientul că

"uită" după ce a fost "interogată". Este foarte important de reţinut că problema poate fi rezolvată cu o memorie foarte simplă care nu suportă decât "o singură citire". Numărătorul nu poate furniza nici o informaţie asupra naturii entităţilor numărate. O soluţie mai comodă pentru a rezolva problemele legate de limbajele independente de context este cea a folosirii memoriei de tip *stivă* (*last in first out, LIFO*). Şi memoria de tip stivă "uită" în momentul în care este citită, dar acest inconvenient nu are importanţă la nivelul limbajelor de tip 2 (va deveni supărător când vom aborda limbajele dependente de context, de tip 1).

Definitia 1.9 *Un automat cu stivă este format prin conectarea unei memorii de tip stivă în bucla unui autoamat finit. Formal:*

$$PDA = (X \times X', Y \times Y' \times X, Q, f, g, z_0)$$

unde:

X *: este alfabetul finit al maşinii; şirul de intrare aparţine mulţimii* X^*

X' *: este alfabetul finit al simbolrilor din stivă,* $X' = X \cup \{z_0\}$

Y *: este mulţimea finită a simbolurilor de ieşire ale maşinii*

Y' *: este mulţimea comenzilor date de automat stivei,* $\{PUSH, POP, -\}$

Q *: este mulţimea finită a stărilor automatului*

f *: este funcţia de tranziţie internă a sistemului:*

$$f : X \times X' \times Q \to Q \times X \times Y'$$

*În funcţia de simbolul recepţionat, de simbolul din vârful stivei (*top of stack*, TOS) şi de starea automatulului finit, automatul comută în altă stare, o nouă valoare este aplicată la intrarea stivei care primeste una din comenzile din* $\{PUSH, POP, -\}$

g *: este funcţei de tranziţie a ieşrii* $g : Q \to Y$

$\mathbf{z_0}$ *: este valoarea iniţială în TOS.*

$\diamond$

Pentru cazurile generale, următoarele două teoreme sunt cunoscute şi demonstrate în toate tratatele de specialitate.

Teorema 1.6 *Orice limbaj de tip 2 poate fi recunoscut prin stările finale ale unui automat cu stivă (*push-down automaton*, PDA).*

◇

Notăm faptul că un PDA este o maşină formată dintr-o structură complexă dar de dimensiune constantă – automatul finit – şi dintr-o structură simplă, definită recursiv, cu dimensiunea dependentă de n – memoria de tip stivă (pentru reîmprospătare de cunoştinţe vizitaţi [39]).

Teorema 1.7 *Orice limbaj de tip 2 poate fi generat de un autoamt cu stivă.*

◇

Acum rămâne să vedem care este diferenţa esenţială dintre un automat finit şi un automat cu stivă. Ce trebuie făcut pentru a obţine o maşină capabilă să recunoască/genereze limbaje independente de context?

Teorema 1.8 *Ordinul minim al unui sistem care implementează un automat cu stivă este 3.*

◇

Demonstraţie Deoarece un automat cu stivă este realizat prin conectarea în buclă a unui autoamt finit cu o memorie de tip stivă, el are un ordin cu o unitate mai mare decât al componentelor conectate în buclă. Automatul finit este un SO2. Memoria de tip stivă este realizabilă în ce mai simplă versiune cu un numărator reversibil (SO2) care adresează o memorie de tip RAM (SO1), care se conectează serial. Deci memoria de tip stivă, în această versiune, este un SO2. În consecinţa, automatul cu stivă este un SO3. Cea de a treia buclă, ce se închide prin stivă are rolul de a memora efectul relaţiei în care se află x cu y ce provin din producţii de tipul $A \to xBy$. Simbolul x este stocat in stivă pentu a fi asociat lui y, atunci când, după un număr de cicluri, acesta din urmă este recepţionat.

◇

Memoria de tip stivă este cea mai simplă formă de memorie (registrul este o stivă cu un singur nivel). Prin definiţie o stivă:

1. stochează şiruri de caractere

2. accesul la şirul de caractere se face la un singur capăt atât pentru scriere cât şi pentru citire (*last - in first - out*)

3. operaţia de citire este distructivă din cauza modului de accesare.

Simplitatea maximă este motivul pentru care acest tip de memorie a fost folosit pentru a concepe maşina care se situează imediat deasupra automatului finit în ierarhia sistemelor digitale. Primul nivel peste nivelul automatului finit este PDA. Dar, acelaşi motiv, al simplităţii, ne va obliga să renunţăm la acest tip de sistem atunci când vom aborda limbajele generate prin producţii context dependente. Vom avea nevoie de memorii al căror conţinut să poată fi accesat de mai multe ori pe parcursul unei proces de recunoaştere. Nu vom mai putea folosi stiva, care "uită" după ce a fost accesată pentru citire. Limbajele dependente de context cer o memorie în care "contextul" să poată fi în mod repetat interogat pentru ca autoamtul finit asociat să poată decide.

1.4.3 Gramatici de tip 1 şi structuri cu 4 bucle (SO4)

Să încercăm folosirea unui automat pentru a aborda măcar unul dintre cele mai simple limbaje din $\mathcal{L}_1$, $\{a^n b^n c^n \mid n \geq 1\}$. Să încercăm chiar folosirea unui automat "infinit". În câteva minute vom decide să abandonăm. Folosind un automat cu stivă, vom ajunge rapid la "soluţia" de a folosi un automat "infinit" pentru a controla stiva. Este evident că trebuie să facem apel la o structură cu o autonomie superioară pentru rezolva problema.

Atunci când vom încerca să folosim pentru recunoaşterea limbajului $\{a^n b^n c^n | n > 0\}$ un automat cu stivă, vom ajunge la concluzia că mai trebuie adăugată o resursă de memorare. Într-adevăr, după ce din stivă au fost citite a-urile, ca urmare a receptării b-urilor, informaţia despre n este pierdută. Numărul de a-uri a fost egal cu cel al b-urilor, dar nu mai avem posibilitatea de a vedea dacă c-urile sunt tot exact la fel de multe. Pentru a compensa această lipsă, o nouă structură de memorare trebuie adăugată într-o *buclă suplimentară*. Vom fi obligaţi astfel să adăugam cea de *a patra buclă*.

În cazul general, va trebui să folosi şi pentru $\mathcal{L}_1$ o structură cu definiţie constantă, independentă de lungimea şirurilor recunoscute sau generate, adică $C_{Machine}(n) \in O(1)$. Această restricţie este satisfăcută dacă vom adăuga în bucla unui automat cu stivă o stivă suplimentară. În acest caz, sistemul devine unul cu 4 bucla de aparţine la SO4. Noua stivă va compensa faptul că citirea stivei iniţiale este distructivă.

Cea de a patra buclă oferă accesul la o memorie suplimentară. Ea a fost necesară datorită faptului că restricţia R2, impusă regulilor de generare, a fost suspendată. Un efect identic se poate obţine prin folosirea în bucla automatului finit a unei memorii cu proprietăţi suplimentare. Este vorba de *memoria linear mărginită*.

Definitia 1.10 *Un automat cu memorie linear mărginită (*linear bounded automaton*, LBA) este un automat finit strict iniţial, conectat în buclă cu o memorie linear mărginită (vezi Figura 1.3) care realizează în fiecare ciclu următoarea secvenţă de operaţii*

1. *generează la ieşirea DOUT a memoriei linear mărginite conţinutul celulei curent accesate*

2. *stochează în celula curent accesată simbolul aplicat la intrarea DIN calculat în funcţie de DOUT şi starea curentă a automatului finit*

3. *accesează pentru ciclul următor (1) celula de memorie din dreapta (prin comanda UP), sau (2) celula din stânga (prin comanda DOWN) sau (3) celula curentă (prin comanda -), implementând astfel o memorie de tip* listă cu acces bidirecţional.

formalizate prin:

$$LBA = (I \cup \{\#\},\ Q, f; q_0)$$

unde: $I \cup \{\#\}$ *este alfabetul finit al maşinii,* Q *este mulţimea finită a stărilor automatului finit,* $q_0 \in Q$ *este starea iniţială a automatului finit, iar* f *este funcţia de tranziţie a întregului sistem:*

$$f = (I \cup \{\#\}) \times Q \rightarrow (I \cup \{\#\}) \times Q \times \{UP,\ DOWN,\ -\}$$

cu restricţia că simbolul # nu poate fi înlocuit în memorie.

În starea iniţială automatul se află în q_0*, şirul asupra căruia se operează se află în memorie limitat la amble capete de simbolul #, iar primul simbol din şir este accesat. În fiecare ciclu, un simbol este citit din memorie de la locaţia curent accesată; automatul determină, în funcţie de acesta şi de starea sa, un simbol care se scrie in memorie la locaţia curent accesată, automatul comută, în funcţie de starea sa şi simbolul curent accesat, în starea următoare.*

◇

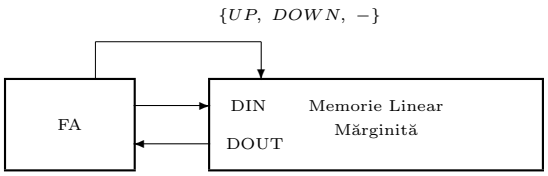


Figura 1.3: Automat cu memorie linear mărginită.

Maşina definită mai sus este folosită pentru generarea sau recunoaşterea limbajelor dependente de context, de tip 1. Procesul de recunoaştere nu poate începe decât dacă secvenţa de recunoscut este în întregime stocată în memoria linear mărginită. Pentru limbajele de tip 3 şi 2, această restricţie nu există. Ea este necesară din cauza dependenţei de context.

Următoarele două teoreme sunt clasice.

Teorema 1.9 *Limbajele dependente de context (de tip 1) sunt recunoscute numai prin stările finale ale unor automate cu memorie linear mărginită (LBA).*

◇

Memoria în care se stochează şirul de recunoscut are celule ce pot fi citite ca locaţii de RAM de ori de câte ori este nevoie, deoarece contextul în care fiecare simbol este generat presupune, la limită, inspectarea întregului şir.

Teorema 1.10 *Limbajele dependente de context sunt generate numai de automate cu memorie linear mărginită (LBA).*

◇

Ambele teoreme sunt demonstrate în toate manualele de limbaje formale. Pentru demersul nostru este nevoie sa demonstrăm următoare teoremă.

Teorema 1.11 *Sistemul care implementează funcţia unui automat cu memorie linear mărginită are cel puţin ordinul 4.*

◇

Demonstraţie Cea mai simplă memorie cu citire nedistructivă se poate construi conectând în buclă două memorii de tip stivă: LIFO 0 şi LIFO 1 (vezi Figura 1.4). Pentru fiecare operaţie de POP0 (extragere din vârful stivei LIFO 0) efectuată în stiva iniţială, se comandă o operaţie de PUSH1 (scriere în vârful stivei LIFO 1) cu conţinutul extras din prima stivă. În acest fel citirea îşi pierde caracterul distructiv. Simetric, pentru orice POP1 din stiva adăugată se poate realiza un PUSH0 în cea iniţială. Adâncimea fiecărei stive poate fi limitată în funcţie de lungimea şirului prelucrat. Cele două stive conectate în buclă realizează funcţia de memorare de tip listă bidirecţională.

Deorece o stivă aparţine SO2, memoria cu citire nedistructivă realizată prin conectarea în buclă a două stive aparţine SO3. Memoria linear mărginită astfel obţinută, conectată în bucla unui automat finit strict iniţial formează, împreună cu acesta din urmă, un SO4 (vezi Figura 1.4).

◇

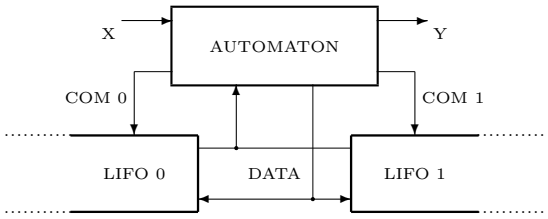


Figura 1.4: Automat cu stivă (AUTOMATON & LIFO 0) conectat în buclă cu stiva adiţională LIFO 1.

O structură echivalentă pentru un LBA este prezentată în Figura 1.5, unde:

- $AUTOMATON$ este un automat finit (un SO2)
- $UDCOUNTER$ este un automat "infinit" cu o structură simplă ($C_{UDCOUNTER} \in O(1)$, cu toate că dimensiunea sa $S_{UDCOUNTER} \in O(logn)$); este folosit ca pointer în memoria RAM
- RAM este o memorie cu acces aleator în care se stochează şirul de simboluri ce se prelucrează într-un spaţiu limitat proporţional cu lungimea şirului (un SO1).

Structura are două bucle ce se includ, închise peste un automat finit. Deci, este un SO4. În soluţia anterioaară şirul se deplasă "prin faţa" automatului.

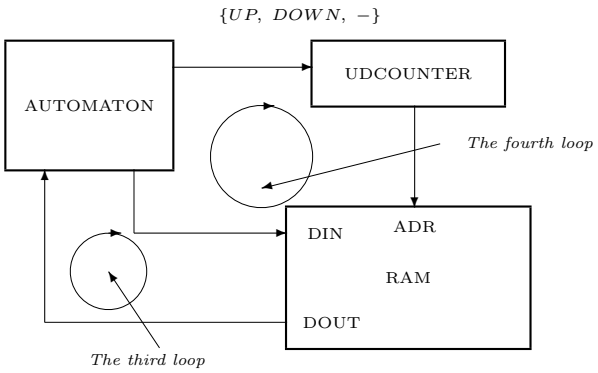


Figura 1.5: Automat cu memorie liniar mărginită.

În această nouă soluţie, conţinutul memoriei este "static" şi este inspectat prin incrementarea sau decrementarea unui numărător folosit ca pointer.

Cerinţele structurale pentru limbajele dependente de context implică o maşină mai complexă, cu funcţii mai net segregate. Cele doua bucle adăugate automatului finit au roluri net distincte:

- prima buclă, închisă prin memoria RAM, este destinată unui *suport de stocare* extern
- cea de a doua, prin $UDCOUNTER$ şi RAM, pentru accesare unei *funcţii de memorare*: *lista bidirecţinal accesată*.

O listă poate face mai multe decât o stivă. Ambele reprezintă un şir, dar stiva permite numai un acces limitat şi destructiv la conţinutul şirului.

1.4.4 Gramatici de tip 0 şi maşini Turing

Modelul de calculabilitate al maşinii Turing [42] este responsabil, împreună cu modelul funcţiilor recursive al lui Kleene [35], pentru impunerea deosebit de imperativă a arhitecturii von Neumann [43] care domină încă (prea autoritar) industria IT&C.

Definitia 1.11 *Maşina Turing (TM) este formată dintr-un automat finit (FA) conectat cu o memorie infinită (vezi Figura 1.6). În fiecare ciclu, automatul realizează următoarele operaţii:*

1. *recepţionează de la ieşirea memoriei, DOUT, conţinutul celulei de memorie curent accesate*

2. *stochează în celula curent accesată un simbol generat pornind de la starea proprie şi de la conţinutul curent al celulei*

3. *selectează celula de memorie pentru ciclul următor ca fiind cea din dreapta celei curente (UP), cea din stânga celei curente (DOWN) sau menţine pe cea curentă (-)*

4. *comută autoamtul în stare următoare, în funcţie de starea curentă şi DOUT.*

Formal scriem:

$$TM = (I,\ Q, f; q_0)$$

unde: I este alfabetul finit al maşinii (simbolurile ce pot fi scrise în memorie), Q este mulţimea finită a stărilor automatului, $q_0 \in Q$ este starea iniţială a automatului finit, iar funcţia f este funcţia de tranziţie a întregii maşini:

$$f = I \times Q \to I \times Q \times \{UP,\ DOWN,\ -\}$$

În fiecare stare, pornind de la simbolul citit din memorie şi de la starea automatului, a nou simbol este scris în memorie, este apoi selecată o nouă celulă de memorie şi automatul comută în starea următoare. În starea iniţială a maşinii Turing automatul este în q_0, memorie conţine şirul de prelucrat delimit la ambele capete de $\# \in I$, iar celula selectată conţine primul element al sirului.

$\diamond$

Diferenţa esenţială dintre *maşina Turing* şi *Automatul cu memorie liniar mărginită* este dată de faptul că la maşina Turing simbolul # poate fi substituit. Consecinţa directă a acestei diferenţe este faptul că nu ştim cât de mare trebuie să fie memoria pentru a putea procesa un şir de lungime dată. Soluţia simplă a acestei probleme a fost să se considere infinit numărul de celule al memoriei maşinii Turing.

Detalierea structurii maşinii Turing este prezentată în Figura 1.7 , unde sunt evidenţiate cele trei componente principale:

1. automatul finit (FA)

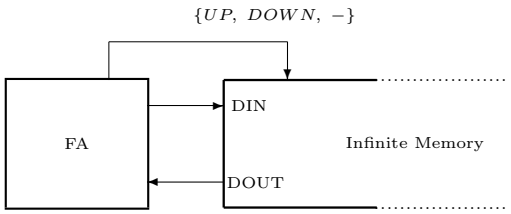


Figura 1.6: Maşina Turing

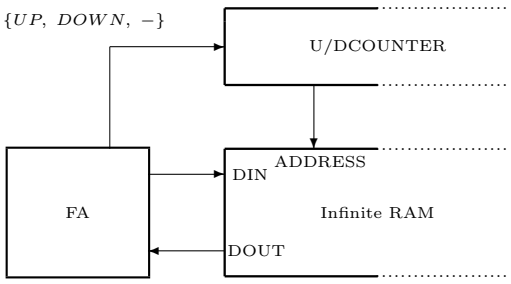


Figura 1.7: Structura maşinii Turing

2. număratorul reversibil, UDCOUNTER, un automat "infinit", un circuit simplu definit recursiv

3. *Memoria infinită*, formată dintr-o memorie RAM adresată prin conţinutul pointerului UDCOUNTER.

Maşina Turing este un concept. O structură fizică corespondentă nu poate fi concepută, datorită caracterului infinit al resurselor de memorie. Nu putem vorbi, deci, despre ordinul sistemului logic care implementează maşina Turing.

Gramaticile de tip 0 şi limbajele asociate sunt caracterizate de reguli de producţie nesupuse niciunor restricţii. Ultima restricţie, R3 (şirul nu poate descreşte în timpul procesului de generare), fiind suspendată, dimensiunea memoriei nu poate fi evaluată, motiv pentru care va fi considerată infinită.

Nota bene: lipsa restricţiilor în definirea regulilor de producţie nu înseamnă că avem de a face cu un limbaj fără restricţii, cu un limbaj natural. Fără restricţii sunt modalităţile de formulare a regulilor. Dar caracterul finit (constant) al dimensiunii mulţimii P a regulilor de producţie

rămâne o restricţie deosebit de puternică. Un limbaj cu $|P|$ constantă este în continuare unul simplu, formal.

1.4.5 Concluzie

Corelaţia dintre *tipurile de limbaj* şi *ordinul* maşinilor fizice asociate este o corelaţie care reflectă relaţia dintre ***expresivitatea*** unui limbaj formal şi ***autonomia*** structurii de circuit asociate, ca maşină de recunoaştere sau generare. Limbajele cu restricţii de generare mai mari vor putea fi recunoscute/generate de structuri de circuit mai puţin autonome, pe când cele cu restricţii mai mici cer structuri de circuit cu autonomie sporită. Nivelul de autonomie al unei structuri de circuit este dat de numărul maxim de bucle care se includ în organizarea circuitului dat. Rezultă corexpondenţa sintetizată după cum urmează:

- $\mathcal{L}_3 \leftrightarrow$ **SO2**
- $\mathcal{L}_2 \leftrightarrow$ **SO3**
- $\mathcal{L}_1 \leftrightarrow$ **SO4**

Avem de a face, în cazul particular al structurilor de recunoaştere/generare a limbajelor formale, cu o ierarhie structurală centrată pe tipul de memorie folosit de un automat finit. Automatul finit, ca SO2, foloseşte cea mai simplă memorie: registrul de stare (sau o stivă cu un singur nivel de stocare). Automatul cu stivă, ca SO3, foloseşte o memorie, cu citire distructivă, de tip stivă, cu o dimensiune dată de ordinul de mărime al şirurilor de recunoscut/generat. Automatul cu memorie liniar mărginită, ca SO4, foloseşte o memorie de tip listă, cu citire nedistructivă.

Caracterul formal al limbajelor considerate permite folosirea unor structuri fizice cu complexitate constantă, chiar dacă dimensiunea (size) acestora poate fi in $O(f(n))$. Componentele simple ale maşinilor asociate limbajelor formale pot fi "infinite", cu condiţia ca să menţină structurile complexe la dimensiuni constante. De aici şi limitele pe cere le au "expresivitatea" limbajelor formale.

1.5 Parsarea

În ştiinţa computaţiei, ***parsarea*** este etapa iniţială a unui interpretor sau a unui complilator. În această etapă, se determină dacă şirul simbolic aparţine limbajului considerat şi care este ***arborele de derivare*** a şirului în gramatica asociată limbajului. Vom prezenta pe scurt, în această secţiune, conceptul de *parsare*. Scopul urmărit este acela de a înţelege esenţa mecanismului pentru a putea folosi în cunoştinţă de cauză instrumente automate de parsare.

Conceptul principal folosit de mecanismul de parsare este arborele de derivare a unui şir aparţinând unui limbaj $L(G)$. Pentru exemplificările din această secţiune vom folosi [40].

Definitia 1.12 *Definim o derivare* ***leftmost*** *(de la stânga la dreapta) independentă de context derivarea lui q din p prin substituţia primului neterminal ce apare în p.*

◇

Definitia 1.13 *Definim o derivare* ***rightmost*** *(de la dreapta la stânga) independentă de context derivarea lui q din p prin substituţia ultimului neterminal ce apare în p.*

◇

Exemplul 1.11 *Fie gramatica*

$$G_7 = (\{S, A\}, \{a, b\}, \{S \to AA,\ A \to AAA|bA|Ab|a\}, S)$$

Urmează, pe două coloane, două derivări distincte care generază acelaşi şir:

```
S -> AA              S -> AA
  -> aA                -> Aa
  -> aAAA              -> AAAa
  -> abAAA             -> AAbAa
  -> abaAA             -> AAbaa
  -> ababAA            -> AbAbaa
  -> ababaA            -> Ababaa
  -> ababaa            -> ababaa
```

Coloana din stânga reprezintă derivarea leftmost *a secvenţei* `ababaa`*, în timp ce coloana din dreapta reprezintă derivarea* rightmost *a aceleiaşi secvenţe.*

◇

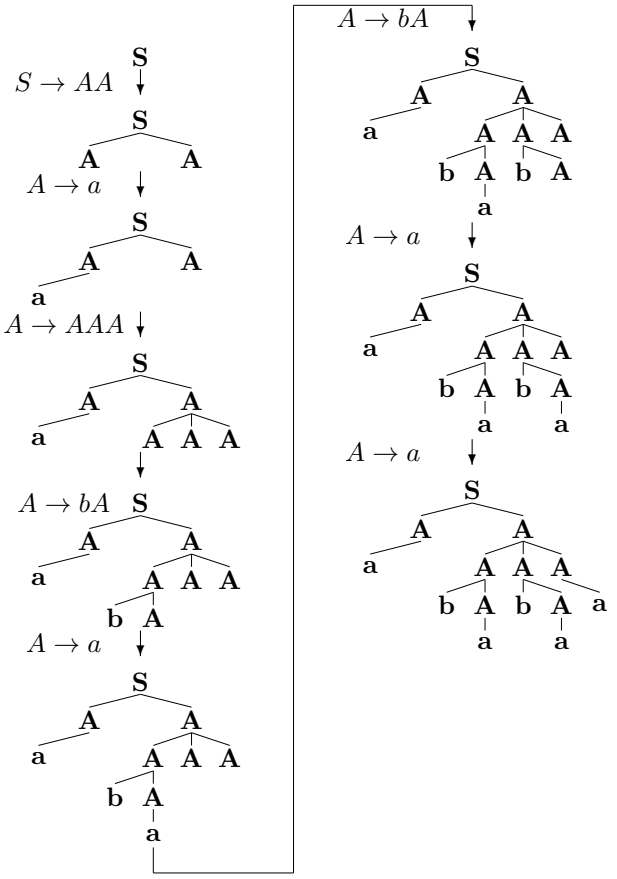


Figura 1.8: Construcţia arborelui de derivare a şirului `ababaa` în gramatica G_7.

Orice secvenţă ce aparţine unui limbaj are asociat un arbore de derivare prin care este definit şirul de producţii care generează secvenţa. Pentru cazul din exemplul anterior, arborele de derivare al secvenţei `ababaa` în G_7 este reprezentat în Figura 1.8, unde fiecare săgeată verticală indică aplicarea producţiei scrise în stânga ei. Prima producţie este $S \to AA$, astfel încât parcurgerea, de la stânga la dreapta, a frunzelor arborelui rezultat reprezintă rezultatul aplicării: AA. A doua producţie, $A \to a$, este marcată în stânga celei de a doua săgeţi şi aduce secvenţa la valoarea `aA`. Se continuă până când frunzele arborelui sunt formate numai din terminale.

Teorema 1.12 *Fie G o gramatică independentă de context. O secvenţă $q \in$*

$L(G)$ dacă şi numai dacă există o derivare leftmost în G a şirului q, pornind din S.

$\diamond$

Demonstraţie : Fie

$$S \to q_1 \to q_2 \to q_3 \to \ldots \to q_{n-1} \to q = S \overset{n}{\underset{G}{\Rightarrow}} q$$

Dacă derivarea este de tip *leftmost*, atunci ne încadrăm în cerinţele teoremei. Dacă nu, atunci există o primă secvenţă, q_{k+1}, care este obţinută printr-o aplicare a unei producţii din P ce nu este *leftmost*. În aceste condiţii numai derivarea parţială $S \overset{k}{\underset{G}{\Rightarrow}} q_k$ este o derivare *leftmost*. Să detaliem secvenţa q_k după cum urmează:

$$q_k = p_1 A p_2 B p_3$$

unde: $p_1 \in T^*$, $p_2, p_3 \in (N \cup T)^*$, A este neterminalul *leftmost* iar B este neterminalul care va fi substituit cu $r \in (N \cup T)^*$ în pasul $k+1$ al derivării printr-o aplicare care nu este de tip *leftmost*, deoarece B este poziţionat la dreapta simboului A, primul neterminal din secvenţă.

Deci, evoluţia derivării iniţiale, care nu este *leftmost*, poate fi fragmentată în 5 etape, după cum urmează:

$$S \overset{k}{\underset{G}{\Rightarrow}} p_1 A p_2 B p_3 \to p_1 A p_2 r p_3 \overset{j}{\underset{G}{\Rightarrow}} p_1 A s \to p_1 p s \overset{n-j-2}{\underset{G}{\Longrightarrow}} q$$

unde: $s, p \in (N \cup T)^*$. Derivarea anterioară, care este *leftmost* în primele k etape, se poate transforma într-o derivare *leftmost* în cel puţin primele $k+1$ etape astfel:

$$S \overset{k}{\underset{G}{\Rightarrow}} p_1 A p_2 B p_3 \to p_1 p p_2 B p_3 \to p_1 p p_2 r p_3 \overset{j}{\underset{G}{\Rightarrow}} p_1 p s \overset{n-j-2}{\underset{G}{\Longrightarrow}} q$$

Transformarea este posibilă pentru că cele două secţiuni ale secvenţei generate în primii k paşi, $p_1 A$ şi $p_2 B p_3$, sunt, în ambele derivări, modificate independent.

Acelaşi procedeu se aplică până când derivarea devine în întregul ei una de tip *leftmost*.

$\diamond$

Teorema anterioară ne va permite să facem ipoteza că aplicarea producţiilor din P se poate face într-o ordine bine stabilită pentru a obţine

un element din $L(G)$. Această condiţie permite simplificarea algoritmilor folosiţi pentru a determina modul de derivare a unui element dat din $L(G)$.

Există două strategii principale pentru a afla care este derivarea şirului de terminale q pornind din simbolul de start S în gramatica G. Prima este aceea a unui ***parser top-down***, care porneşte din S în arborele de derivare asociat gramaticii G în tentativa de a găsi şirul q. Cea de a doua strategie este a unui ***parser bottom-up***, care porneşte de la şirul q, poziţionat în frunzele arborelui de derivare şi caută calea către simbolul de start S. De asemenea, parcurgerea arborelui de derivare se poate face în două feluri: *breath-first* sau *depth-first*. Pentru scopul pe care-l urmărim, vom opta pentru a prezenta numai un singur algoritm de parsare. Optăm, din considerente de simplitate, pentru algoritmul de parsare *depth-first top-down*.

1.5.1 Algoritmul de parsare *depth-first top-down*

Algoritmul de parsare *depth-first top-down* construieşte arborele de derivare al şirului de simboluri parsate, dacă acesta există, pornind cu simbolul de start al gramaticii considerate. Regulile de derivare ale gramaticii sunt numerotate pentru a se stabili o ordine de aplicare. În fiecare nod al arborelui este aplicată regula ce îndeplineşte următoarele trei condiţii: (1) aparţine submulţimi celor aplicabile; (2) nu a mai fost aplicată în acel nod; (3) are indexul minim. Procesul avansează numai dacă în fiecare etapă prefixul delimitat de primul neterminal este un prefix al şirului parsat.

Algoritmul (vezi Figura 1.9) presupune folosirea unei memorii de tip stivă pentru a controla procesul de parsare.

Vârful stivei (top of stack) `tos = [left, right]` conţine o pereche formată dintr-un şir $left = (N \cup T)^* \cup \{\#\}$ şi un număr $right = i$. Simbolul $\#$ este folosit pentru a iniţializa conţinutul stivei, iar numarul i, când $i \neq 0$, reprezintă indexul ultimei reguli de producţie aplicate în încercarea de a găsi o producţie acceptată de procesul de parsare. Şirul `left` reprezintă forma pe care o are şirul "propus" de generea în etapa anterioară.

Variabila `deadEnd` este `false`, atâta timp cât procesul de parsare avansează "promiţător"; atunci când prin aplicarea unei reguli de producţie se obţine un prefix de terminale ce nu se regăseste ca prefix în secvenţa de parsat, variabila `deadEnd` ia valoarea `true`.

Şirul $p = uAv$ reprezintă stadiul curent al tentativei de a genera şirul de parsat. Simbolul A reprezintă primul neterminal din p, $u \in T^* \cup \{\lambda\}$, iar $v \in (N \cup T)^* \cup \{\lambda\}$

Procesul de parsare este format dintr-o două bucle ce se includ. Bucla internă aplică primului neterminal regula cu indexul minim posibil, dar mai mare de `tos[right] = i`, dacă acesta există. Bucla se repetă atât timp cât procesul nu generează un prefix format din terminale ce nu aparţine ca prefix şirului de parsat.

```
procedure  depthFirstTopDownParser
   push(#,0); initializarea stivei, (tos= #,0))== stiva golita
   push(S,0);
   loop
      deadEnd <= false;
      p <= tos[left];
      i <= tos[right];
      pop;
      loop
         if (not(u == prefix(q)))            deadEnd <= true;
         if (Rj: A -> w, j>i does not exist) deadEnd <= true;
         if (deadEnd == false) { push(p, j);
                                 p <= uwv;
                                 i <= 0;
                               }
      until (deadEnd == true) | p format numai din terminale
   until ((p == q) | (tos == (#, 0))
   if (p == q) ACCEPTAT;
    else     REJECT;
endprocedure
```

Figura 1.9: Algoritmul `depthFirstTopDownParser`

Bucla principală descarcă (`pop`) din stivă: (1) rezultatul unei etape de generare ce s-a dovedit că nu conduce către secvenţa de parsat şi (2) indexul regulii de generare ce s-a dovedit a fi aplicată greşit. Cu noile valori pentru p şi i, se reintră în bucla internă. Procesul continuă până când şirul p coincide cu şirul q, a cărui generare este investigată, sau până c2nd toate încercarile de a găsi un şir de producţii ce conduce de la S la q se dovedesc zadarnice. Imposibilitatea de a parsa un şir este dată de `tos = (#, 0)`, testată condiţionat de `deadEnd = true`.

Dacă parsarea este reuşită, atunci din stivă se vor putea extrage, în ordine inversă, din componentele `right` ale fiecărei înregistrări, indexurile regulilor

de generare ce trebuie aplicate pornind din S pentru a obţine şirul de parsat q.

1.5.2 Exemplu de parsare

Din considerente de simplitate, fie o gramatică, G_A, care generează expresii aditive. G_A este suficient de simplă pentru a exemplifica uşor mecanismul de parsare *depth-first top-down*, descris în paragraful anterior.

Definitia 1.14 *Fie gramatica G_A:*

$$G_A = (N, T, P, S)$$

unde:

```
N = {S, A, T}
T = {a, +, (, )}
P = {S -> A     ,   (1)
     A -> T     ,   (2)
     A -> A+T   ,   (3)
     T -> a     ,   (4)
     T -> (A)   }   (5)
```

Cele cinci producţii au fost numerotate pentru a putea fi apelate în descrierea algoritmică folosind indexul asociat.

◇

Expresiile generate cu G_A sunt de tipul: `a`, `(a+a)`, `(a+a+a)`, `(a+(a+a))`, `(a+(a+(a+a)))`, `(a+(a+(a+a))+a)`, ş.a.m.d. Spre exemplu, arborele de derivare al expresiei `(a+a)`, folosind gramatica G_A, este reprezentat în Figura 1.10. Pornind din S, sunt aplicate, în ordinea crescătoare a indexului asociat, regulile ce se pot aplica într-o modalitate de derivare *leftmost*. Dacă în fiecare pas prefixul delimitat de primul neterminal este un prefix al expresiei de parsat, atunci suntem pe calea cea bună, altfel efectul pasului "greşit" este anulat şi se încearcă regula cu indexul imediat superior, dacă acest lucru este posibil.

Expresia `(a+a)` fiind una ce aparţine limbajului $L(G_A)$, arborele de parsare se poate obţine, iar secvenţa de reguli a căror aplicare conduce la

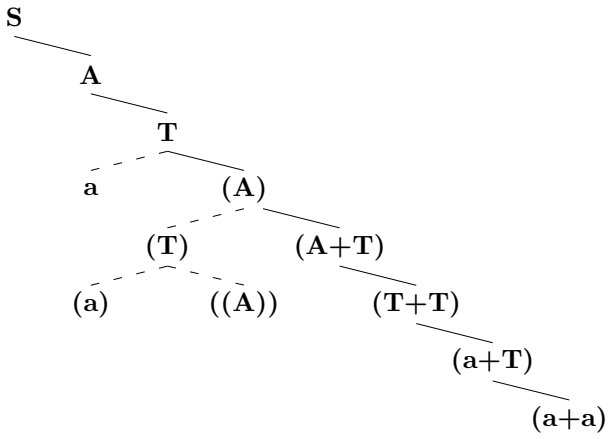


Figura 1.10: Calea de derivare a şirului `(a+a)` în gramatica G_A investigată prin algoritmul de parsare *depth-first top-down*. Căile redate cu linie întreruptă reprezintă încercările nereuşite.

`(a+a)` reprezintă rezultatul procesului de parsare. În Figura 1.10, este reprezentat schematic procesul de parsare al expresiei `(a+a)`. Conform algoritmului `depthFirstTopDownParser`, procesul porneşte din rădăcina arborelui iniţiată la valoarea simbolului de start `S`. Se aplică unica regulă posibilă, (1), şi se generează nodul `A`. În această etapă, pot fi aplicate două reguli, (2) şi (3). O vom aplica pe cea cu indexul mai mic şi rezultă nodul `T`. Şi în această etapă putem aplica două reguli. Conform algoritmului descris, o vom aplica, pentru început, pe cea cu index mai mic, (4). Rezultă nodul `a`. Dar `a` nu se regăseşte ca prefix în şirul `(a+a)`. Ne întoarcem în nodul `T` şi căutăm o altă regulă aplicabilă. Aceasta este regula (5), şi ajungem în nodul (A). În şirul deja format `u = (` este un prefix ce aparţine şirului de parsat, deci putem continua. Aplicarea regulilor (2) şi (4) se dovedeşte nepotrivită, pentru că ea conduce la şirul `(a)`. Înlocuirea regulii (4) cu (5) este iarăşi "neinspirată", motiv pentru care ne întoarcem în nodul `(A)` şi aplicăm regula (3). Din acest punct, aplicarea algoritmului conduce direct, fără `pop`-urile marcate prin linii punctate în Figura 1.10, la soluţia căutată.

Pentru a înţelege modul în care este folosită stiva în procesul de parsare descris, vom relua funcţionarea algoritmului `depthFirstTopDownParser`

pentru expresia `(a+a)` vizualizând în Figura 1.11 evoluţia procesului de derivare prin variabilele implicate în algoritm:

```
STEP     0     1     2     3     4     5     6     7     8     9

deadEnd: -     -     -     false false false false true  false false
tos:     [...] [#,0] [S,0] [#,0] [S,1] [A,2] [T,4] [A,2] [T,5] [(A),2]
               [...] [#,0] [...] [#,0] [S,1] [A,2] [S,1] [A,2] [T,5]
                     [...]       [...] [#,0] [S,1] [#,0] [S,1] [A,2]
                                       [...] [#,0] [...] [#,0] [S,1]
                                             [...]       [...] [#,0]
                                                               [...]

i:       -     -     -     0     0     0     0     4     0     0
p:       -     -     -     S     A     T     a     T     (A)   (T)


10      11      12      13      14    15      16        17        18

false   true    false   true    true  false   false     false     false
[(T),4] [(A),2] [(T),5] [(A),2] [T,5] [(A),3] [(A+T),2] [(T+T),4] [(a+T),4]
[(A),2] [T,5]   [(A),2] [T,5]   [A,2] [T,5]   [(A),3]   [(A+T),2] [(T+T),4]
[T,5]   [A,2]   [T,5]   [A,2]   [S,1] [A,2]   [T,5]    [(A),3]   [(A+T),2]
[A,2]   [S,1]   [A,2]   [S,1]   [#,0] [S,1]   [A,2]    [T,5]     [(A),3]
[S,1]   [#,0]   [S,1]   [#,0]   [...] [#,0]   [S,1]    [A,2]     [T,5]
[#,0]   [...]   [#,0]   [...]         [...]   [#,0]    [S,1]     [A,2]
[...]           [...]                         [...]    [#,0]     [S,1]
                                                        [...]     [#,0]
                                                                  [...]

0       4       0       5       2     0       0         0         0
(a)     (T)     ((A))   (T)     (A)   (A+T)   (T+T)     (a+T)     (a+a)
```

Figura 1.11: Evoluţia stării procesului de parsare asociat expresiei `(a+a)` din $L(G_A)$.

- `deadEnd`: variabila care indică ajungerea într-o "fundătură", din care se poate ieşi extrăgând din stivă pasul greşit care a condus la un prefix ce nu apare în aceaşi calitate în secvenţa de parsat

- conţinutul stivei în care se stochează "urma" procesului de parsare

- indexul `i` care delimitează inferior indicele regulii ce poate fi aplicată în etapa curentă a parsării: orice regulă R_j aplicată trebuie să îndeplinească condiţia $i < j$
- şirul `p` reprezintă şirul propus în fiecare etapă a procesului de parsare ca un posibil candidat pe calea ce conduce la generearea şirului de parsat.

Etapele parsării ilustrate în Figura 1.11 sunt:

STEP 0 : stare iniţială în care stiva nu conţine nimic semnificativ, iar `deadEnd`, `i` şi `p` au valori nesemnificative

STEP 1 : este iniţializată stiva în starea "golită"

STEP 2 : `push [S,0]`, iniţializarea rădăcinii arborelui de derivare în `S`, iar zeroul semnifică faptul că orice regulă de producţie poate fi aplicată

STEP 3 : se intră în bucla principală prin extragerea din `tos` a "stadiului" în care a juns parsarea: nodul curent conţine `S` şi se poate aplica orice regulă pentru ca `i = 0`

STEP 4 : se intră în bucla interioară prin:

- `push [S,1]`: şirului `S` i s-a aplicat regula de producţie (1)
- a rezultat noua formă a şirului: `A`
- resetarea indexului (`i = 0`): noii forme a lui `p` i se poate aplica oricare dintre regulile ce-i sunt asociate

STEP 5 : se mai parcurge odată bucla interioară pentru că `deadEnd = false` şi se execută:

- `push [A,2]`: şirului `A` i s-a aplicat regula de producţie (2)
- a rezultat noua formă a şirului `T`
- resetarea indexului (`i = 0`)

STEP 6 : se mai parcurge odată bucla interioară pentru că `deadEnd = false` şi se execută:

- `push [T,4]`: şirului `T` i s-a aplicat regula de producţie (4)
- a rezultat noua formă a şirului `a`

- resetarea indexului (`i = 0`)

STEP 7 : se detectează că `a` nu este un prefix al şirului de parsat şi se iese din bucla internă în cea principală prin

- `p = left[tos] = T`
- `i = right[tos] = 4`
- `pop`

STEP 8 : se reintră în bucla interioară şi se execută:

- `push [T,5]`: şirului `T` i s-a aplicat regula de producţie (5), după ce s-a dovedit că aplicarea regulii cu index inferior, 4, nu conduce către secvenţa *parsată*
- a rezultat noua formă a şirului `(A)`
- resetarea indexului (`i = 0`)

...

STEP 18 : elementele `right` din înregistrările din stivă reprezintă şirul de reguli de producţie căutat listat în ordine inversă. Deci şirul de reguli identificat este: `(1), (2), (5) (3), (2), (4), (4)`.

Programele care operează parsarea sunt curent realizate prin actualizarea unor generatoare de parsare ce sunt iniţializate prin "încărcarea" gramaticii limbajului din care se face traducerea. Un astfel de program este, spre exemplu, ANTLRWorks (ANTLR ANother Tool for Language Recognition), un mediu pentru dezvoltarea de gramatici scris de Jean Bovet (vezi Anexa A).

Capitolul 2

Aplicaţii muzicale ale gramaticilor

Folosirea procedeelor enuntate de gramaticile generativ-transformaţionale, în ocurenţă, gramaticile de tip 2, independente de context (*context-free grammars*), se face cu urmărirea a două obiective:

- definirea unui vocabular şi a unui set de reguli în cazul concret al unui corpus de lucrări muzicale clasice (în ocurenţă, sonatele de maturitate ale lui W. A. Mozart) ;

- apoi, definirea unui nou vocabular înlocuitor al celui original şi, mai ales, a unui nou set de reguli, derivabile, prin diverse procedee transformaţionale, din cele iniţiale.

Aceste obiective sunt, la rîndul lor, subordonate unui *target* mai mare, respectiv deducerea unei gramatici generalizate ce înglobează coordonate ale limbajului muzical mozartian.

Noam Chomsky însuşi a formulat anumite direcţii ce ar putea privi cercetarea de faţă, atunci cînd şi-a propus, sistem(at)ic, întrebarea dacă muzica poate fi inclusă în teritoriul limbajelor (fie precum cele naturale, fie în categoria limbalelor formale). Rezerva sa, principială, se purta asupra faptului că muzica este un *limbaj* lipsit de referenţialitate.

Luăm în considerare posibilitatea reală ca muzica să fie omologată unui tip special de limbaj formal. Pentru a-i putea detecta, cum spuneam, virtualităţile formatoare de structuri.

În acelaşi timp, este de subliniat faptul că proiectul acesta vizează convertirea unui tip de discurs sonor (în ocurenţă, cel teleologic), într-unul de tip deschis. Acelaşi Noam Chomsky vorbeşte de caracterul special al muzicii, pe care o denumeşte "pseudo-limbaj", întrucît – spune el – muzica nu are un *referent* precum limbile naturale. Dac luăm în consideraţie celebrul triunghi al lui Pierce, pe care îl regăsim şi la Ferdinand de Saussure, avem relaţia complexă dintre semnificant, semnificat şi referent (vezi Figura 2.1).

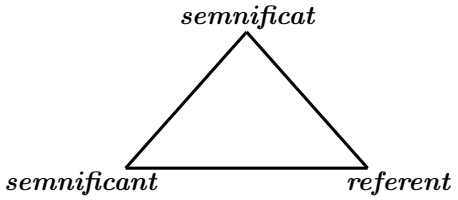


Figura 2.1: Triunghiul lui Pierce.

Muzica, însă, nu aparţine nici zonei limbajelor formale (precum matematica ori logica formală). Tocmai acest statut intermediar este cel care interesează în studiul de faţă. Situaţia primă a utilizării gramaticilor generative se sprijină pe asimilarea în prealabil, a diferitelor aspecte pe care le propun, în general, aceste gramatici. În acest scop, au fost făcute expunerile anterioare.

Pentru cercetarea de faţă, am ales un anumit protocol de desfăşurare, ale cărui etape sunt derulate precum urmează:

Etapa I : alegerea unui subset al sonatelor mozartiene de pian, pentru a fi folosit la identificarea unei gramatici sursă

Etapa II : identificarea gramaticii sursă

Etapa III : definirea unor gramatici destinaţie numai prin modificarea terminalelor

Etapa IV : definirea unor gramatici destinaţie prin extensii gramaticale sau noi gramatici

Etapa V : traducerea unui fragment mozartian în limbajele generate de gramaticile definite într-una din cele două etape anterioare.

2.1 Etapa I: Identificarea spaţiului sonor sursă

Au fost alese, pentru început, trei Sonate pentru pian de Wolfgang Amadeus Mozart, din perioada de maturitate a creaţiei sale: KV 331, KV 332 şi KV 570 (vezi Figura 2.2, unde sunt reproduse numai primele măsuri din fiecare sonată). De aici, mai întâi, vom decupa frazele muzicale ale mîinii drepte. În continuare, se desprind exclusiv structurile ritmice. (De subliniat faptul că, pentru claritatea expunerii, s-a procedat la o omogenizare a valorilor: în ocurenţă, doimea a fost înlocuită de pătrime, pătrimea de optime etc. Aceasta pentru claritatea expunerii şi uşurarea travaliului). S-a obţinut, astfel, un număr restrîns de valori ritmice.

Primele 4 măsuri din sonata KV 331 se reduc la schema ritmică reprezentată în Figura 2.3.

Primele 7 măsuri din sonata KV 332 se reduc la schema ritmică reprezentată în Figura 2.4.

Primele 6 măsuri din sonata KV 570 se reduc la schema ritmică reprezentată în Figura 2.5.:

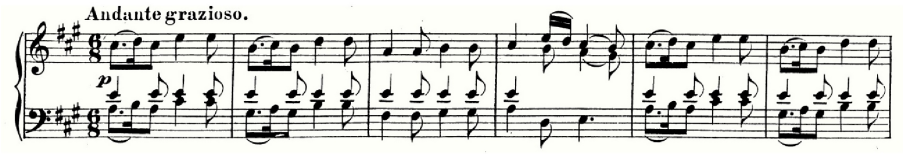
Pentru omogenitatea şi claritatea demersului, vom considera în sonatele KV 332 şi KV 570, că valorile reale sunt reduse la jumătate.

2.2 Etapa II: Extragerea gramaticii sursă

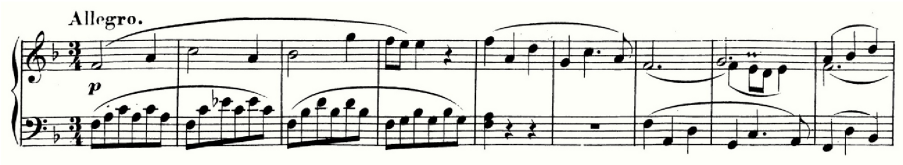
Vom admite următoarea convenţie:
şaisprezecime = 1;
optime = 2;
optime cu punct = 3;
pătrime = 4;
pătrime cu punct = 6.

Se impun, aici, cîteva observaţii, vizînd acest vocabular (pentru că este un vocabular al celor trei exemple alese):

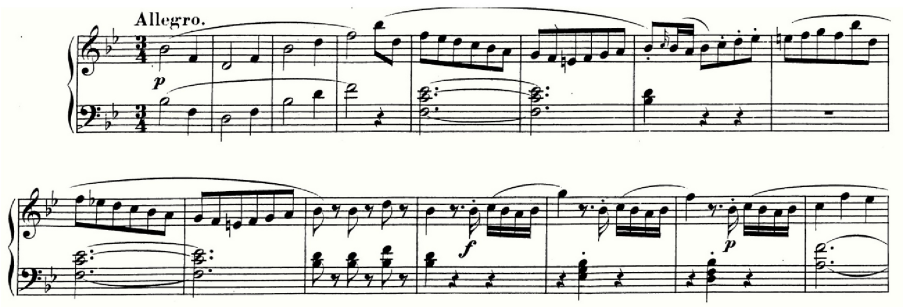
- din şirul 1, 2, 3 , 4, 6 numerele 1, 2, 4 reprezintă puteri ale lui 2 (2^0, 2^1, 2^2), Iar numerele 3 şi 6, reprezintă următoarea formulă: $2^n + 2^{n-1}$. Evident, acest şir poate fi făcut să crească indefinit, atât la dreapta, cât şi la stânga.



Fragment din sonata KV 331



Fragment din sonata KV 332



Fragment din sonata KV 570

Figura 2.2: **Fragmente din sonate de Mozart**

Figura 2.3: **Primele măsuri din sonata KV 331**

Figura 2.4: **Primele măsuri din sonata KV 332**

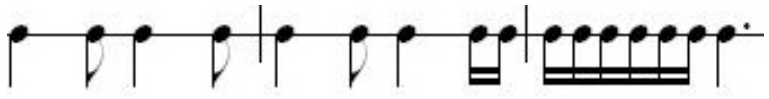


Figura 2.5: **Primele 4 măsuri din sonata KV 570**

- puterile lui 2 pot creşte la "infinit", în sens pozitiv, dar şi negativ. Iar din 2 la puterea n, poate fi scăzută valoarea 2 la puterea $n1$ plus 2 la puterea $n2$. Etc.

Dăm, în continuare, schemele numerice ale celor trei sonate.

```
4    2   4  2  4    2  4  2   |   4  2  4  2  4  2    4  2                (1)
3 1  2   4  2  3 1  2  4  2   |   4  2  4  2  4  1 1  4  2
```

```
4 2  4 2  4 2  2   4   |   2 4    2   4    2 4  2 4                       (2)
4 2  4 2  4 2  1 1 4   |   2 2 2  2   3 1  6    6
```

```
4 2  4 2  4 2  4 2   | 4       2    4        2    4       2    4       2  (3)
4 2  4 2  4 2  4 1 1 | 1 1 1 1 1 1  1 1 1 1  1 1  1 1 1 1 1 1  1 1 1 1 1 1
```

Linia verticală reprezintă axa de simetrie a fragmentului.

Următoarea gramatică:

$$G_{M1} = (\{S, A, B, C\}, \{1, 2, 3, 4, 6\}, P, S)$$

unde: P conţine următoarele reguli de producţie
$S \to ASA \,|\, AA$
$A \to BC \,|\, CB \,|\, 6$
$B \to 4 \,|\, 31 \,|\, 22 \,|\, 1111$ $C \to 2 \,|\, 11$
este "extrasă" din primele trei fragmente mozartiene.

Un prim exemplu de derivare va genera chiar primul eşantion (1) folosit în definirea gramaticii:
S
$\to ASA$
$\to AASAA$
$\to AAAAAA$
$\to BCAAAAA$

...
$\rightarrow BCBCBCBCBCBC$
$\rightarrow 31CBCBCBCBCBC$
$\rightarrow 312BCBCBCBCBC$
$\rightarrow 3124CBCBCBCBC$
$\rightarrow 31242BCBCBCBC$
$\rightarrow 3124231CBCBCBC$
$\rightarrow 31242312BCBCBC$
...
$\rightarrow 31242312424CBC$
$\rightarrow 3124231242411BC$
$\rightarrow 312423124241 14C$
$\rightarrow 312423124241142$

Un al doilea exemplu va genera o structură înrudită cu primul, dar neprovenind din nici o situaţie concretă a lucrărilor de Mozart. Pornind din S generăm:

S
$\rightarrow ASA$
$\rightarrow AASAA$
$\rightarrow AAAAAA$
$\rightarrow BCAAAAA$
$\rightarrow BCBCAAAA$
$\rightarrow BCBCCBAAA$
$\rightarrow BCBCCBCBAA$
$\rightarrow BCBCCBCBBCA$
$\rightarrow BCBCCBCBBC6$
$\rightarrow 4CBCCBCBBC6$
$\rightarrow 42BCCBCBBC6$
$\rightarrow 424CCBCBBC6$
$\rightarrow 4242CBCBBC6$
$\rightarrow 424211BCBBC6$
$\rightarrow 4242114CBBC6$
$\rightarrow 42421142BBC6$
$\rightarrow 4242114231BC6$
$\rightarrow 42421142314C6$
$\rightarrow 4242114231426$

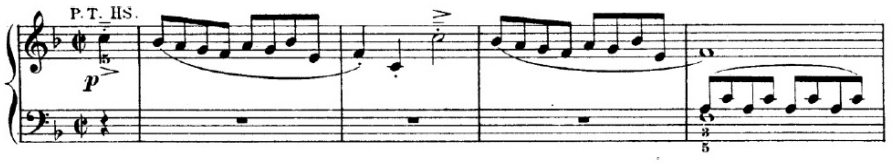
Este evident că secvenţa rezultată:

4 2 4 2 1 1 4 2 3 1 4 2 6

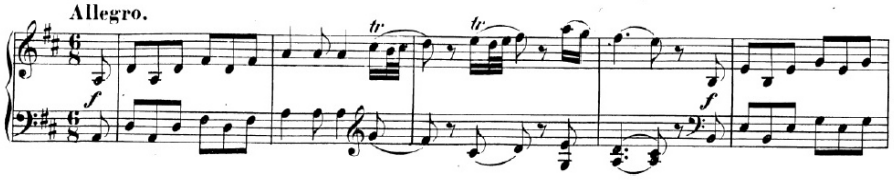
este congruentă cu prima, cu alte cuvinte, aparţine modalităţilor mozartiene de a crea structuri ritmice.

Este de remarcat, aici, următorul aspect: în ceea ce priveşte vocabularul, pentru definirea acestuia în mod eficient, trebuie utilizate cel puţin cupluri de valori. De exemplu: 42, sau 31 etc.

În continuare, problema incitantă ce se pune, ar fi descoperirea, prin alte gramatici asemănătoare, a unor lanţuri ce ar corespunde altor fraze muzicale din alte sonate mozartiene. Vom considera exemplele din Figura 2.6.



Fragment din sonata KV 494



Fragment din sonata KV 576

Figura 2.6: **Fragmente din sonate de Mozart folosite pentru validarea gramaticilor.**

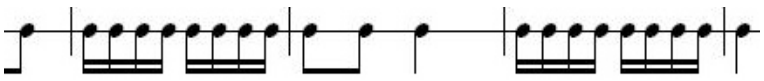


Figura 2.7: **Primele 4 măsuri din sonata KV 494**

Prin aplicarea unui tratament identic celui aplicat eşantioanelor din Figura 2.2, rezultă secvenţle din Figura 2.7 şi din Figura 2.8. Interesant ar fi de văzut în ce măsură gramatica G_{M1} poate fi folosită pentru a genera secvenţele anterioare.

Figura 2.8: **Primele 4 măsuri din sonata KV 576**

2.3 Etapa III: Definirea alfabetului ţintă

O altă problemă - de fapt, esenţială pentru configurarea proiectului de faţă - este cea a definirii vocabularului destinaţie. Acest lucru se poate face prin extensia vocabularului sursă sau prin propunerea unui alt vocabular. Concret, în cazul utilizării gramaticilor, este vorba despre definirea terminalelor gramaticii destinaţie.

Luăm în considerare două modalităţi de redefinire:

- prin folosirea unor algoritmi simpli şi reversibili de transformare, pornind de la alfabetul clasic
- prin folosirea unor algoritmi complecşi şi ireversibili de transformare pornind de la acelaşi alfabet clasic.

2.3.1 Algoritmi reversibili

Folosirea unor valori rar utilizate

Să considerăm, dintr-un exemplu anterior, celula din Figura 2.9 pe care o înlocuim cu, mult mai rara, formulă din Figura 2.10.

Figura 2.9:

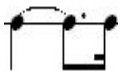


Figura 2.10:

Într-adevăr, în limbajul ritmurilor mozartiene această formulă este atipică, putând fi întâlnită doar în mod excepţional, ca unicat.

Din punct de vedere matematic, această extensie de limbaj, prin extensie de vocabular, înseamnă folosirea unor valori obţinute prin însumarea unor puteri ale lui 2, ca în exemplul următor:

$$2^n + 2^{n-1} + 2^{n-2}$$

Prin această substituţie, vocabularul ritmic extras din Mozart este prelungit cu formule aparţinând limbajelor altor compozitori (din alte perioade stilistice). Se produce implicit o "spărtură" în omogenitatea limbajului de la care am pornit.

Folosirea unor valori excepţionale

Din acelaşi exemplu propunem o substituire mai marcantă: valorile egale de optimi din Figura 2.11 – şase la număr – sunt înlocuite cu trei grupuri, după cum urmează: triolet de optimi, cvintolet de şaisprezecimi, triolet de optimi (vezi Figura 2.12).

Figura 2.11:

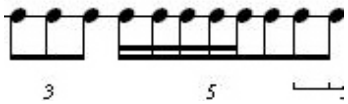


Figura 2.12:

Din punct de vedere matematic, această extensie înseamnă folosirea unor valori de timp ce cad în intervalul dintre unele puteri ale lui 2:

$$2^n < 2^n + 1 < 2^{n+1}$$

Ruptura astfel realizată este mult mai mare, mai "spectaculoasă" faţă de obişnuinţele ritmice mozartiene. Se intră în zona, mult ulterioară, a romanticilor şi post-romanticilor.

Folosirea unor serii neconvenţionale

Seria ritmică folosită de Mozart (de toţi clasicii) este numită convenţională pentru că se bazează, în principal, pe puterile lui doi. Aceasta înseamnă că, pornind dela o valoare foarte mare (de exemplu, nota întreagă), prin împărţiri succesive cu 2, ajungem la o valoare extrem de mică, 128-imea. Considerând că nota întreagă se obţine prin împăţirea cu 2^0, ultima valoare se obţine prin împăţirea cu 2^7. Seria de numere folosită pentru a diviza nota întreagă este seria puterilor cu 2. Folosirea unor *serii neconvenţionale* este o modalitate profitabilă de a extinde alfabetul gramaticii ţintă. Vom exemplifica prin folosirea seriei Fibonacci şi a unor serii aleatoare.

Seria Fibonacci: Prin contrast cu datele clasice, o serie ritmică bazată, să spunem, pe binecunoscutul şir al lui Fibonacci – $1, 1, 2, 3, 5, 8, 13, 21, \ldots$ – format dintr-o secvenţă de numere în care fiecare număr este egal cu suma celor două anterioaare. O proprietate a acestui şir este aceea că raportul a două numere succesive tinde către $1,618$ – rădăcina pozitivă a ecuaţiei $x^2 - x - 1 = 0$ –, valoare cunoscută şi sub numele de *secţiune de aur*.

Şirul lui Fibonacci poate fi considerat o serie neconvenţională, aproape neîntâlnită în muzica numită, generic, clasică. Atunci, seriei ritmice anterioare, construită prin divizarea cu puterile lui 2, i se va stubstitui o serie configurată prin împărţirea cu elementele sirului Fibonacci (din care excludem a doua apariţie a lui 1). Rezultă următoarea corespondenţă, pornind de la seria convenţională:

$$\begin{aligned} \textit{notă întreagă}\ 1/2^0 &\rightarrow 1/1\ \textit{notă întreagă} \\ \textit{doime}\ 1/2^1 &\rightarrow 1/2\ \textit{doime} \\ \textit{pătrime}\ 1/2^2 &\rightarrow 1/3\ \textit{doime de triolet} \\ \textit{optime}\ 1/2^3 &\rightarrow 1/5\ \textit{pătrime de cvintolet} \\ \textit{şaisprezecime}\ 1/2^4 &\rightarrow 1/8\ \textit{optime} \\ &\ldots \end{aligned}$$

Remarcăm următoarele lucruri:

- din când în când, elementele celor două şiruri coincid (mai ales în prima parte a seriei)
- şirul fibonaccian apare ca fiind mai neregulat în prima parte a seriei

- şirul fibonacian tinde, pentru valori mari, către o regularitate similară cu cea a seriei convenţionale, cu deosebirea că raportul de 2 dintre două componente este înlocuit cu raportul de 1,618.

Rezultă şi un al doilea mod de folosire: seria convenţională a puterilor lui 2 este înlocuită cu seria puterilor *secţiunii de aur*. Va rezulta o serie de timp ce nu va putea fi executată decât de instrumente simulate computaţional, pentru un public ce va trebui să accepte o tranziţie culturală semnificativă.

Serii aleatoare O variantă extremă de generare a seriilor ritmice (frecvenţiale sau de altă natură) este folosirea unor şiruri de numere aleatoare. Astfel, pornind de la 1, se însumează un şir de numere generate aleator pentru a se constitui seria neconvenţională.

Spre exemplu, dacă se porneşte de la seria de numere aleatoare: $(3, 8, 5, 13, 2, 17)$ se obţine seria temporală neconvenţională aleatoare:

$$(1, 4, 12, 17, 30, 32, 49)$$

Pentru folosirea seriei ritmice rezultate avem două soluţii. Execuţia controlată de computer, care poate opera cu intevale de timp realizate cu o acurateţe ce depăşeşte uşor capacitatea umană de a discernere. O a doua posibilitate este de a rotunji valorile aleator obţinute prin valori convenţionale sau prin valori ce pot fi găsite într-o serie neconvenţinală mai puţin complexă (de tip Fibonacci, spre exemplu).

Nota bene: Am prezentat regulile de construcţie a seriilor numerice pe care se bazează structurarea alfabetelor muzicale folosite (cu exemplificări date numai în domeniul ritmurilor). Aceste reguli au generat serii din ce în ce mai complexe. Este o tendinţă normală ce sprijină creşterea expresivităţii discursului muzical. Dar utilizarea seriilor neconvenţionale trebuie făcută cu moderaţia care permite creşterea complexităţii discursului muzical numai în limitele în care obiectul sonor nu se apropie periculos de zona purului aleator, de zona "zgomotului" ce exclude total orice formă de sintaxă, semnificaţie sau sens.

Sintaxa conferă ceorenţă internă, semnificaţiile păstrează o conexiune cu spaţiul cultural în care discursul muzical are loc, pe când sensul muzical transcende orice referent, constituind astfel "conţinutul" ultim al demersului muzical. Sensul se sprijină pe *redundanţa* dată de semnificaţiile întrupate în

paradigme culturale ce nu pot fi excluse complet. O altă formă de manifestare a *redundanţei* este dată de elementele sintactice care nu trebuie să rămână complet transparente receptorului.

2.3.2 Algoritmi ireversibili

În această secţiune, ne vom abate de la domeniul ritmului muzical, pe care l-am abordat până acum , pentru a trece în acela al frecvenţelor. Considerăm că, pentru ceea ce vom prezenta în continuare, este mai convenabil acest procedeu. Transformările ireversibile prin care se poate construi un alfabet ţintă vor fi exemplificate în domeniul frecvenţelor.

Exemplul 2.1 *Fie organismul sonor reprezentat de spectrul de armonice impare*

$$[1^0, 3^0, 5^0, \ldots, 31^0] = [16, 5Hz, 49, 5Hz, 82, 5Hz, \ldots, 511, 5Hz]$$

al unei fundamentale, fie aceasta $16,5\,Hz$*. Transformarea pe care o avem în vedere constă din:*

- *generarea unui spectru intermediar ce conţine numai componente realizate prin însumarea unui număr impar de componente, selectate aleator, ale spectrului iniţial; spre exemplu:* $[(1^0+5^0+11^0), (3^0+7^0+13^0+1^0+5^0+17^0+21^0), \ldots]$
- *spectrul intermediar obţinut este transformat în continuare prin împărţirea componentelor sale cu puteri întregi, alese aleator, ale lui 2.*

Rezultatul final este spectrul următor:

$$[140, 25Hz, 264Hz, \ldots]$$

Se observă faptul că spectrul obţinut este inarmonic, în sensul că posedă componente care sunt multipli neîntregi ai fundamentalei.

Se observă, de asemenea, că spectrul obţinut este ireversibil, în sensul că nu putem indica modul în care a fost obţinut.

◇

Pentru a obţine ***restrângerea*** domeniului în care parametrii organismului sonor iau valori, la un interval dat, (min, max), propunem completarea exemplului anterior dupa cum urmează.

Exemplul 2.2 *Fie organismul sonor reprezentat de spectrul de armonice impare*

$$[1^0, 3^0, 5^0, \ldots, 31^0] = [16,5Hz, 49,5Hz, 82,5Hz, \ldots, 511,5Hz]$$

al unei fundamentale, fie aceasta $16,5\,Hz$. *Transformarea pe care o avem în vedere constă din:*

- *generăm un spectru intermediar ce conţine numai componente realizate prin însumarea unui număr impar, p, de componente, stabilit aleator, ale spectrului iniţial; spre exemplu:* $[(1^0 + 5^0 + 11^0), (3^0 + 7^0 + 13^0 + 1^0 + 5^0 + 17^0 + 21^0), \ldots]$
- *spectrul intermediar obţinut este transformat în continuare prin împărţirea componetntelor sale cu* 2^p, *unde p a fost asociat, în primul pas, fiecărei componente a acestui spectru intermediar*
- *din spectrul obţinut se păstrează numai componentele din intervalul de parametri stabilit de la început, fie acesta, pentru exemplul nostru:* $(min, max) = (20Hz, 120Hz)$

Rezultatul etapei a 2-a este spectrul următor:

$$[((1+5+11)/2^3)\times 16,5Hz, ((3+7+13+1+5+17+21)/2^7)\times 16,5Hz, \ldots] =$$

$$[35,0625Hz, 11,2148Hz, \ldots]$$

iar în etapa a 3-a reţinem numai:

$$[35,0625Hz, \ldots]$$

$\diamond$

Un al doilea exemplu, care conduce la rezultate asemănătoare, se bazează pe metoda concepută de acusticianul şi muzicianul John Chowning [16] [17]. În substanţă, Chowning construieşte spectre de inarmonice ce conţin frecvenţe de forma:

$$f_n = f_0 \pm f_m$$

unde: f_0 este frecvenţa purtătoare (modulată), iar f_m este frecvenţa modulantă.

Exemplul 2.3 *Fie f_0 o frecvenţă aflată la jumătatea ambitusului dintre cea mai joasă frecvvenţă (fundamentala) şi cea mai înaltă care a fost aleasă în spectrul iniţial din exemplul anterior (31^0). Rezultă că f_0, semnalul modulat, va lua o valoare în jurul frecvenţei $(31^0 - 1^0)/2 = (30 \times 16, 5Hz)/2 = 247, 5$, care reprezintă exact armonicul 15^0.*

Semnalul modulator trebuie să fie obligatoriu un inarmonic în raport cu f_0. Pentru a fi siguri, vom lua o frecvenţă din spectrul de inarmonice construit în exemplul anterior. Fie aceasta $f_m = 35, 0625Hz$. Spectrul rezultat va fi generat de relaţia:

$$f_n = 247, 5Hz \pm n \times 35, 0625Hz$$

unde n este un întreg pozitiv. Rezultă spectrul:

$$[\ldots 177, 37Hz,\ 212, 43Hz,\ \mathbf{247, 5Hz},\ 282, 56,\ 317, 625Hz, \ldots]$$

care este limitat la stânga la valori mai mari decât $16, 5Hz$ – pragul frecvenţelor audibile – iar la dreapta de plafonul spectrului utilizabil.

În aplicaţiile concrete, spectrul obţinut poate fi supus unei limitări (filtrări) prin care o serie de componente sunt eliminate.

◇

2.4 Etapa IV: Definirea regulilor de producţie ţintă

Revenim, pentru cele ce urmează, la domeniul valorilor, adică la domeniul ritmurilor. Definirea unei gramatici ţintă se împlineşte ca funcţie prin definirea mulţimii regulilor de producţie. Acest lucru se poate obţine în mai multe moduri:

- generarea prin extinderea setului de reguli
- generarea prin însumarea unor seturi de reguli
- generarea unor multimi de reguli ce conţin perechi regulă-probabilitate (gramatici markoviene)

Cele trei moduri anterior enunţate pot fi folosite ca atare sau mixate.

2.4.1 Extinderea unei gramatici

Ne vom referi numai la obţinerea unei noi gramatici prin mărirea numărului de reguli. O implicaţie, în majoritatea cazurilor obligatorie, este mărirea mulţimii neterminalelor şi terminaleor.

Exemplul 2.4 *Fie gramatica definită prin următoarea mulţime de producţii:*

$$P = \{S \to Aa, A \to Ba, B \to b|Bb\}$$

O extindere posibilă este următoarea:

$$P = \{S \to Aa, A \to Ba|Aa, B \to b|Bb\}$$

la care am adăugat posibilitatea generării unui număr oricât de mare de a-uri.

◇

Exemplul 2.5 *Fie gramatica definită prin următoarea mulţime de producţii:*

$$P = \{S \to Aa, A \to Ba, B \to b|Bb\}$$

O extindere posibilă este următoarea:

$$P = \{S \to Aa, A \to Ba, B \to b|Bb|BC, C \to Cc|c\}$$

în acest caz, mulţimea iniţială a neterminalelor – $\{S, A, B\}$ – s-a îmbogăţit devenind $\{S, A, B, C\}$ – $\{S, A, B\}$ –, iar mulţimea teminalelor – $\{a, b\}$ – a devenit – $\{a, b, c\}$.

◇

Acest tip de extindere nu afectează esenţa structurilor din limbajul ţintă. Propune numai "ramificaţii" suplimentare.

2.4.2 Însumarea gramaticilor

Un procedeu mai complex de generare a unei gramatici ţintă este cel de *însumare a gramaticilor*. Procedeul constă din reuniunea mulţimilor ce caracterizează gramaticile "însumate".

Exemplul 2.6 *Fie gramaticile:*

$$G_1 = (\{S, A\}, \{a, b\}, \{S \to aA, A \to aA|b\}, S)$$

$$G_1 = (\{S, B\}, \{b, c\}, \{S \to b|bB, B \to cB|c\}, S)$$

Suma lor va fi:

$$G = (\{S, A, B\}, \{a, b, c\}, \{S \to aA|b|bB, A \to aA|b, B \to cB|c\}, S)$$

$\diamond$

Efectul scontat este generarea unor structuri lingvistice cu valenţe expresive lărgite. Procedeul trebuie aplicat condiţionat de faptul că gramaticile "însumate" nu pot fi complet incongruente între ele.

2.4.3 Gramatici probabiliste

Până în acest moment, probabilitatea aplicării unei reguli era identică pentru toare elementele mulţimii P. Dacă, spre exemplu, cardinalul lui P era 5, atunci toate regulile aveau o ocurenţă egală cu probabilitatea 1/5. În cele ce urmează, vom lua în considerare cazul mai realist în care regulile de producţie sunt aplicate cu probabilităţi inegale. Adică, mulţimea P este definită ca o mulţime de perechi (r_i, p_i), unde r_i reprezintă o regulă de producţie, iar p_i probabilitatea de a fi aplicată.

Exemplul 2.7 *Să reluăm gramatica anterior construită pentru eşantioanele mozartiene alese:*

$$G_{M1} = (\{S, A, B, C\}, \{1, 2, 3, 4, 6\}, P, S)$$

unde: P conţine următoarele reguli de producţie:
$S \to ABA$
$A \to AACA$
$B \to BC \,|\, CB$
$C \to CD$
$A \to 42$
$B \to 31$
$C \to 1111$
$D \to 2222$

Observaţia pe care o putem face, reexaminând eşantioanele considerate, este că aceste reguli sunt aplicate cu ocurenţe inegale. Putem propune, cu aproximaţie, următoarea asociere probabilistică, reformulând mulţimea P sub formă de perechi (producţie, probabilitate), după cum urmează:
$(S \to ABA)$
$(A \to AACA(0,6) \,|\, 42(0,4))$
$(B \to BC(0,2) \,|\, CB(0,5) \,|\, 31(0,3))$
$(C \to CD(0,45), \,|\, 1111(0,55))$
$(D \to 2222)$

$\diamond$

Iniţial am vorbit despre gramatici în care aplicarea producţiilor se putea face echiprobabil. Aceasta oferea o libertate care scotea rezultatele de sub orice implicare tehnico-stilistică. Din aceste raţiuni am introdus un prim nivel de restricţii. Fără acesta, rezultatul generării s-ar dispersa într-un spaţiu mult prea extins.

Probabilităţile asociate se pot stabili în cel puţin două feluri:

- funcţie de materialul ales pentru stabilirea gramaticii sursă, prin raportare la rezultatele obţinute prin parsare
- funcţie de strategia decizională aleasă prin poziţionarea faţă de gramatica sursă

2.4.4 Probleme

În această secţiune, se propun câteva exerciţii simple de aplicare a proceddelor anterior descrise.

Problema 2.1 *O primă aplicaţie se referă la cele două sonate suplimentare - KV 494 şi KV 576. Iată, translatarea numerică a structurilor celor două partituri (e vorba de prima frază - mâna dreaptă – a lor):*

```
2 1 1 1 1  1 1 1 1  2 2   4 1 1 1 1  1 1 1 1  6        (4)


2  2 2 2  2 2 2   4  2   2   4  1 1  4 1 1  4  1 1  6  (5)
```

Se observă, imediat, că vocabularul rămîne acelaşi.

Încercaţi parsarea acestor două secvenţe, folosind gramatica anterior definită. În cazul în care nu reuşiţi, propuneţi modificări până când parsarea reuşeşte.

◇

Problema 2.2 *Cele trei exemple de la început (sonatele KV 331, 332, 570) vor fi transformate în structuri ritmice diferite, prin folosirea procedeelor prezentate în secţiunea 2.3.1 Algoritmi reversibili.*

◇

2.5 Etapa V: Traducerea în limbajul ţintă

Pentru început, vom traduce unul din fragmentele, alese anterior, pentru definirea gramaticii mozartiene, aşa cum am evidenţiat-o folosind eşantioanele selectate în etapa I.

În continuare, vom considera un eşantion suplimentar, deja prezentat în 2.2, pentru a-l traduce în aceeaşi gramatică ţintă.

2.5.1 Traducerea fragmentului din KV 332

Pornim de la gramatica extrasă în Etapa II. O reamintim:

$$G_{M1} = (\{S, A, B, C\}, \{1, 2, 3, 4, 6\}, P, S)$$

unde: P conţine următoarele reguli de producţie numerotate pentru a facilita procesul de parsare:

$R1: \ S \to ASA$
$R2: \ S \to AA$
$R3: \ A \to BC$
$R4: \ A \to CB$
$R5: \ A \to 6$
$R6: \ B \to 4$
$R7: \ B \to 31$
$R8: \ B \to 22$
$R9: \ B \to 1111$
$R10: \ C \to 2$

$R11: \ C \to 11$

Pentru a obţine cea mai simplă translatare, vom defini o gramatică nouă realizată numai prin extinderea mulţimii P, după cum urmează:

$$G_{M2} = (\{S, A, B, C\}, \{1, 2, 3, 4, 6\}, P, S)$$

unde: P conţine următoarele reguli de producţie:
$D1: \ S \to ASA$
$D2: \ S \to AA$
$D3: \ A \to BC$
$D4: \ A \to CB$
$D5: \ A \to 6$
$D6: \ A \to 111111$
$D7: \ A \to 21111$
$D8: \ B \to 4$
$D9: \ B \to 22$
$D10: \ B \to 31$
$D11: \ B \to 22$
$D12: \ B \to 1111$
$D13: \ C \to 2$
$D14: \ C \to 11$
Se observă imediat că au fost extinse numai regulile cere generau substituţia finală a terminalelor, în acest fel *pattern*-ul de bază nu este "atacat" organic.

Corespondenţa dintre mulţimile P_1 şi P_2 este următoarea:

$$R1 \Rightarrow D1$$
$$R2 \Rightarrow D2$$
$$R3 \Rightarrow D3$$
$$R4 \Rightarrow D4$$
$$R5 \Rightarrow D5 \,|\, D6 \,|\, D7$$
$$R6 \Rightarrow D8 \,|\, D9$$
$$R7 \Rightarrow D10$$
$$R8 \Rightarrow D11$$
$$R9 \Rightarrow D12$$
$$R10 \Rightarrow D13$$
$$R11 \Rightarrow D14$$

Se poate estima că, prin această extensie, vom obţine ramificaţii de aceeaşi natură cu cele de la care s-a pornit, însă lărgite şi/sau îmbogăţite.

Vom alege pentru translatare structura ritmică a primelor 7 măsuri din sonata KV 332 reprezentate în Figura 2.13:

Figura 2.13: **Primele măsuri din sonata KV 332**

Reprezentarea lor numerică este următoarea:

```
4 2  4 2  4 2  11 4  |  2 22  2 31  6
```

Vom realiza următoarea secvenţă de operaţii asupra şirului anterior:

1. parsarea după gramatica G_{M1}, din care rezultă şirul de reguli R_1
2. determinarea şirului de reguli corespunzătoare în G_{M2} – R_2
3. aplicarea şirului R_2 simbolului de start, pentru obţinerea rezultatului traducerii.

Parsarea

Folosind programuul *ANTLRWorks* din Anexa 1, rezultă următoarea secvenţă de reguli aplicate în G_{M1}, pornind din S pentru a genera şirul de tradus:

```
R1 R1 R1 R2 R3 R3 R3 R4 R3 R4 R6 R10 R6 R10 R6 R01 R11 R6
R8 R10 R10 R7 R5 R5
```

Pentru a verifica corectitudinea şirului de reguli obţinut, le vom aplica simbolului de start S şi vom obţine următoarea secvenţă de generare:

```
R1:    ASA
R1:    AASAA
R1:    AAASAAA
R2:    AAAAAAAA
R3:    BCAAAAAAA
R3:    BCBCAAAAAA
R3:    BCBCBCAAAAA
R4:    BCBCBCCBAAAA
```

```
R3:     BCBCBCCBBCAAA
R4:     BCBCBCCBBCCBAA
R6:     4CBCBCCBBCCBAA
R10:    42BCBCCBBCCBAA
R6:     424CBCCBBCCBAA
R10:    4242BCCBBCCBAA
R6:     42424CCBBCCBAA
R10:    424242CBBCCBAA
R11:    42424211BBCCBAA
R6:     424242114BCCBAA
R8:     42424211422CCBAA
R10:    4242421142222CBAA
R10:    4242421142222BAA
R7:     42424211422223 1AA
R5:     4242421142222316A
R5:     42424211422223166
```

Construirea listei de reguli corespondente

Folosind corespondenţa, anterior stabilită, între regulile de tip Ri şi Dj, obţinem următorul tabel:

```
R1:  D1
R1:  D1
R1:  D1
R2:  D2
R3:  D3
R3:  D3
R3:  D3
R4:  D4
R3:  D3
R4:  D4
R6:  D8 (selectat din D8 | D9)
R10: D13
R6:  D9 (selectat din D8 | D9)
R10: D13
R6:  D8 (selectat din D8 | D9)
R10: D13
```

```
R11: D14
R6:  D9 (selectat din D8 | D9)
R8:  D11
R10: D13
R10: D13
R7:  D10
R5:  D7 (selectat din D5 | D6 | D7)
R5:  D5 (selectat din D5 | D6 | D7)
```

Generarea şirului tradus

```
D1 : ASA
D1 : AASAA
D1 : AAASAAA
D2 : AAAAAAAA
D3 : BCAAAAAAA
D3 : BCBCAAAAAA
D3 : BCBCBCAAAAA
D4 : BCBCBCCBAAAA
D3 : BCBCBCCBBCAAA
D4 : BCBCBCCBBCCBAA
D8 : 4CBCBCCBBCCBAA
D13: 42BCBCCBBCCBAA
D9 : 4222CBCCBBCCBAA
D13: 42222BCCBBCCBAA
D8 : 422224CCBBCCBAA
D13: 4222242CBBCCBAA
D14: 422224211BBCCBAA
D9 : 42222421122BCCBAA
D11: 4222242112222CCBAA
D13: 42222421122222CBAA
D13: 422224211222222BAA
D10: 4222242112222224AA
D7 : 4222242112222224211111A
D5 : 422224211222222421111 6
```

2.5.2 Traducerea unor fragmentelor suplimentare

Reprezentarea numerică a unuia din cele două fragmente (KV 494) este:

```
2 | 1111 1111 | 224 | 1111 1111 | 6
```

Se observă două aspecte care nu pot fi generate de gramatica G_{M1}:

- prezenţa *auftakt*-ului
- simetria manifestată cu un centru actual, nu virtual, ca în primele trei exemple folosite pentru a defini gramatica. Este vorba de un grup BC sau CB care este, tocmai, *axă de simetrie*.

În virtutea acestor observaţii, apare necesitatea completării gramaticii G_{M1}, după cum urmează:

$$G_{M2} = (\{S, A, B, C, D\}, \{1, 2, 3, 4, 6\}, P, S)$$

unde: P conţine următoarele reguli de producţie:
$R1: \ S \to DASA$
$R2: \ S \to ASA$
$R3: \ S \to AA$
$R4: \ S \to A$
$R5: \ A \to BC$
$R6: \ A \to CB$
$R7: \ A \to 6$
$R8: \ B \to 4$
$R9: \ B \to 31$
$R10: \ B \to 22$
$R11: \ B \to 1111$
$R12: \ C \to 2$
$R13: \ C \to 11$
$R14: \ D \to 1$
$R15: \ D \to 2$
Am adăugat următoarele reguli de producţie:

$S \to DASA$ pentru a permite parsarea unei secvenţe ce începe cu *auftakt*

$S \to A$ pentru a permite un centru actual de simetrie

$D \to 1 \,|\, 2$ pentru a concretiza *auftakt*-ul.

Cu aceste adaosuri, considerăm că ambele eşantioane suplimentare pot fi parsate cu succes.

2.6 Concluzii

Demersul expus pînă acum are următoarea logică: structurile unui discurs omologat de către istoria muzicii (în ocurenţă, sonatele pentru pian solo de W. A. Mozart, din care au fost decupate doar structurile ritmice) au fost analizate în vederea stabilirii unor gramatici generative de tip doi, în vederea extinderii acestora prin serii de vocabular şi de sintaxă cât mai cuprinzătoare, mai îndepărtate de modelul original.

Cum am mai arătat, prelevarea paradigmatică a textelor alese realizează stabilirea alfabetului, vocabularului, apoi stabilrea regulilor sintagmatice - gramatica - şi punctul de start. Apoi, se pregăteşte un nou vocabular şi un nou set de reguli.

Schema procesului este următoarea:

- analiza textului clasic (în ocurenţă, sonatele pentru pian de Mozart)
- stabilirea vocabularului acestuia
- se regăseşte un vocabular nou şi un nou set de reguli. Vocabulatul V devine vocabularul V'
- setul reguli G devine setul reguli G'
- limbajul sursă $L(V, G, S)$ devine limbajul destinatie $L'(V', G', S')$.

Capitolul 3

Fractali

Fractalii reprezintă o încercare de a stăpâni complexitea reală prin mijloace algoritmice ce iau în considerare numai o complexitate aparentă. Abordarea de tip fractal mai porneşte de la ipoteza că realitatea posedă şi o foarte fină organizare ierarhică ce conservă o aceeaşi structură la toate nivelurile. Studiul fractal conservă astfel două "obsesii" ale culturilor abrahamice (mozaică, creştină şi islamică): *simplitatea profunzimilor* (absolute şi abstracte) şi organizarea ierarhică de tip "*as above, so below*"[1].

În acest capitol, vom porni de la definirea *complexităţii* şi a *complexităţii aparente* pentru a putea trata fractalii drept o modalitate de a aproxima complexitatea reală prin cea aparentă, atunci când acest lucru este posibil sau util. Demersurile din categoria creaţiei de tip artistic pot beneficia de teoria fractalilor care permite controlul complexităţii în vederea adaptării ei la capacitatea umană de creaţie, interpretare şi percepţie.

3.1 Complexitate

Complexitatea este un concept la fel de dificil de definit ca timpul, forma, spaţiul, informaţia şi multe alte concepte pe care le folosim cu nonşalanţă, cu toate că nu posedăm definiţii riguroase unanim acceptate. Ne vom mulţumi, din acest motiv, cu măsura complexităţii, care este mai uşor de definit. O vom folosi mai apoi pentru a defini, aproape riguros, ce înţelegem printr-o realitate complexă.

[1] *Emerald Tablet.*

3.1.1 Complexitatea algoritmică

În deceniul 7 al secolului al XX-lea, Ray Solomonoff [34], Andrey Nikolaevich Kolmogorov [24] şi Gregory Chaitin [4] au definit în mod independent ***complexitatea algoritmică***. Redăm mai jos, după o lucrare a lui Chaitin [5], prin câte o întrebare sintetică, propunerile celor trei matematicieni referitor la ideea de măsură a complexităţii.

Solomonoff se întreabă: *cum se poate construi o teorie pornind de la un şir de măsurători realizate într-un experiment desfăşurat asupra unei relităţi riguros delimitate?* Solomonof evidenţiază două soluţii limită:

- în şirul de date obţinut se poate pune în evidenţă un anumit pattern ce va fi declarat ca reprezentând legea ce guvernează procesul investigat; cercetătorul poate publica o lucrare în care comunică pattern-ul descoperit.

- în şirul de date obţinut nu se poate pune în evidenţă un anumit patern, caz în care nu se poate identifica, din prspectiva datelor disponibile, o legitate ce ar guverna procesul supus investigaţiei; cercetătorul poate publica o lucrare, dacă i se acceptă, o lucrare în care listează şirul de date obţinut prin experimentul desfăşurat, cu speranţa că altcineva va fi în stare să evidenţieze un pattern.

Kolmogorov se întreabă: *care este diferenţa dintre următoarele două şiruri binare, de lungime n, echiprobabile într-un experiment prin care sunt determinate aleator?*

secventa_1: `01001011101010010101101001011010...`
secventa_2: `01100110011001100110011001100110...`

Teoria clasică a probabilităţii, la care Kolmogorov avusese contribuţii majore, statuează faptul că cele două şiruri de biţi apar, într-un experiment "condus" aleator, cu probabilităţi egale. Şi totuşi, la 62 de ani distinsul savant are sentimentul că cel de al doilea şir pare a avea o caracteristică ce nu poate fi ignorată; măcar pentru faptul că poate fi memorat cu uşurinţă, în comparaţie cu primul care necesită un efort special. Teoria la care contribuise decisiv trebuia completată cu posibilitatea de a distinge în astfel de situaţii.

Pentru cel de al doilea şir binar Kolmogorov consideră, în [24], că există o funcţie ce-l poate genera, spre deosebire de primul şir pentru care o astfel de funcţie nu poate fi definită.

Chaitin se întreabă: *care este cel mai scurt program care generază o anumită secvenţă binară?* În cazul celor două secvenţe binare anterioare, rezultă următoarele programe. Pentru `secventa_1` programul cel mai scurt este următorul:

```
print: 0100101110101001010110100101101O...
```

Lungimea programului rezultat este proporţională cu cea a şirului generat, pentru că suntem obligaţi să listăm explicit în program şirul de generat. Pentru `secventa_2` programul cel mai scurt este următorul:

```
print n times: 0110
```

Lungimea programului rezultat este proporţională cu logaritmul din lungimea şirului generat, deoarece valoarea n este scrisă cu un număr de simboluri ce este proporţional cu $log\ n$.

Chaitin conchide: complexitatea secvenţei `secventa_1` este mare şi dependentă de dimensiunea n a şirului generat, în timp ce complexitatea secvanţei `secventa_2` este mult mai mică, proporţională cu $log\ n$. Formal, scriem:

$$C_{secventa\_1} \in O(n)$$

$$C_{secventa\_2} \in O(log\ n)$$

în timp ce dimensiunile (size) celor două secvenţe sunt egale şi aparţin aceluiaşi ordin de mărime, după cum urmează:

$$S_{secventa\_1} = S_{secventa\_2} \in O(n)$$

Cred că suntem acum pregătiţi pentru a încerca definiţii riguroase referitor la complexitate.

Definitia 3.1 *Complexitatea algoritmică a unei realităţi date este proporţională cu lungimea celei mai scurte descrieri (a celui mai scurt program) care poate genera acea realitate.*

◇

Intuitiv, trebuie să acceptăm faptul că o realitate este cu atât mai complexă cu cât cea mai mică descriere a acestei realităţi este mai concisă. Se demonstrează faptul că nu există un procedeu formal care să ne garanteze în orice caz real faptul că am ajuns la ceea mai scurtă descriere simbolică. În funcţie de simbolurile şi de limbajul folosit, măsura complexităţii poate varia. Acesta este motivul pentru care, în definiţia anterioară, am folosit expresia *este proporţională* în loc de *este*.

Dacă ne referim la descrierea printr-o procedură de generare, întrupată într-un program, este evident faptul că vom estima complexitatea unei realităţi oarecare prin lungimea, exprimată în număr de caractere, a unui program.

Definitia 3.2 *O realitate este simplă dacă are dimensiunea în $O(f(n))$ şi complexitatea algoritmică în $O(g(n))$, unde funcţia $f(n)$ creşte cel puţin polinomial, iar funcţia $g(n)$ creşte cel mult logaritmic.*

◇

Conform acestei definiţii, pentru un $n > n_0$, realitatea R are întotdeauna dimensiunea mult mai mare decât complexitatea. Adică:

$$S_R(n) >> C_R(n)$$

pentru $n > n_0$, dacă R este simplu.

Definitia 3.3 *O realitate este complexă dacă are dimensiunea în $O(f(n))$ şi complexitatea algoritmică în $O(g(n))$, unde funcţiile $f(n)$ şi $g(n)$ cresc cel puţin polinomial.*

◇

Rezultă că, pentru o realitate complexă, R, este îndeplinită condiţia:

$$S_R(n) \sim C_R(n)$$

pentru $n > n_0$. Dacă "povestea" care descrie realitatea R este realizată cu un "text" având o lungime care este în acelaşi ordin de mărime cu dimensiunea realităţii R, atunci R este complexă.

O tablă de şah are un desen ce poate fi descris printr-o explicaţie foarte scurtă, pe când un tablou al lui Dali presupune o poveste ce se poate întinde şi până dincolo de răbdarea ascultătorilor. Vom fi deci acord că o tablă de şah este simplă, pe când un tablou de Dali este complex.

Vom evita, astfel, o serie de confuzii, încă prezente în manuale, dintre *size* – $S(n)$ – şi *complexity* – $C(n)$. $S(n)$ se referă la *dimensiunea realităţii*, pe când $C(n)$ se referă la *dimensiunea descrierii* acelei realităţi. În ambele cazuri avem de a face cu măsuri ce se pot exprima cantitativ.

Dar, ***ce este complexitatea?*** Această întrebare rămâne o întrebare deschisă! Ca şi în cazul timpului, pe care-l măsurăm cu precizii incredibil de mari, în cazul complexităţii avem posibilitatea de a o măsura, fără a şti cu rigurozitate *ce* măsurăm. Trebuie să ne mulţumim cu sugestiile metaforice ale unui cuvât din ce în ce mai folosit. Poate că expresivitatea şi, prin aceasta, utilitatea lui constă tocmai în caracterul lui metaforic, pe care-l menţinem, inconştient poate, din raţiuni subtile legate de felul în care mentalul nostru funcţionează.

3.1.2 Complexitatea aparentă

Realităţile pe care reuşim să le descriem formal nu se încadrează strict în două categorii, cea a relităţilor simple şi cea a realităţilor complexe. Se mai poate evidenţia o formă de complexitate diferită. Nu neapărat intermediară. Este vorba despre realităţi a căror complexitate depinde de "punctul de vedere" din care este investigată. Este vorba despre ceea ce vom numi realităţi *aparent complexe*.

Complexitatea aparentă caracterizează realităţi cu următoarele proprietăţi:

- sunt realităţi generate folosind reguli descrise prin secvenţe constante sau cel mult cu dimensiune logaritmică
- regula de generare nu poate fi dedusă, sau nu poate fi dedusă uşor, din analiza formală a realităţii generate.

Complexitatea aparentă caracterizează realităţi simple care au o aparenţă complexă pentru cei ce nu cunosc regula de generare. Mulţi consideră că întreaga realitate este aparent complexă, datorită faptului că *încă* nu am reuşit să descoperim regulile simple care o generează sau guvernează. Existenţa multor situaţii, în care complexitatea se dovedeste ca fiind într-adevăr aparentă, face ca această opinie să domine spaţii culturale vaste.

Vom exemplifica cu câteva cazuri în care, într-adevăr, nivelul de complexitate efectiv este inaccesibil celor aflaţi pe poziţia de spectatori. "Păpuşarii" ştiu, însă, cât de mică este complexitatea reală.

Exemplul 3.1 *Fie configuraţia de biţi din Figura 3.1 care este generată de un sistem digital a cărui ieşire a fost iniţializată la valoarea* $0^{58}10^{58}$. *În fiecare ciclu de evoluţie ieşirea de 117 biţi se modifică. Secvenţa binară obţinută are o evoluţie foarte complexă, în sensul că nu poate fi pus în evidenţă uşor mecanismul intern de generare. Comportamentul aparent neperiodic ne sugerează o complexitate mare. Însă, pentru cine cunoaşte mecanismul de generare, lucrurile devin foarte simple. Într-adevăr, secvenţa din Figura 3.1 este generată de un* automat celular *linear cu structura şi funcţia descrise de următorul cod Verilog:*

```
module linearCellularAutomaton #(parameter n = 117)
       (output reg [n-1:0] out ,
        input  [n-1:0] in ,
        input     init, clock);
   always @(posedge clock)
    if (init) out        <= in                                        ;
     else     out[n-2:1] <= (out[n-2:1] | out[n-3:0]) ^ out[n-1:2];
endmodule
```

Funcţia implicată este o simplă funcţie logică de trei varaibile

$$(A + B) \oplus C$$

care este efectuată în fiecare element al unei structuri celulare (pentru detalii se va consulta [39] secţiunea 11.2). Codul, de dimensiune constantă (fiene permis să ignorăm faptul că pentru a exprima valoarea parametrului `n` *folosim un număr de simboluri care creşte logaritmic), ce apare între* `module` *şi* `endmodule` *descrie o matrice bidimensională de biţi ce se poate extinde oricât de mult.*

În aplicaţii industriale, o coloană cu lăţimea de m biţi poate fi considerată, cu bună aproximaţie, o secvenţă aleatoare de numere reprezentate pe m biţi.

◇

Exemplul 3.2 *Vom relua exemplul anterior cu o modificare minoră la nivelul definiţiei:*

```
module linearCellularAutomaton #(parameter n = 95)
  (output reg [n-1:0] out ,
   input  [n-1:0] in ,
```

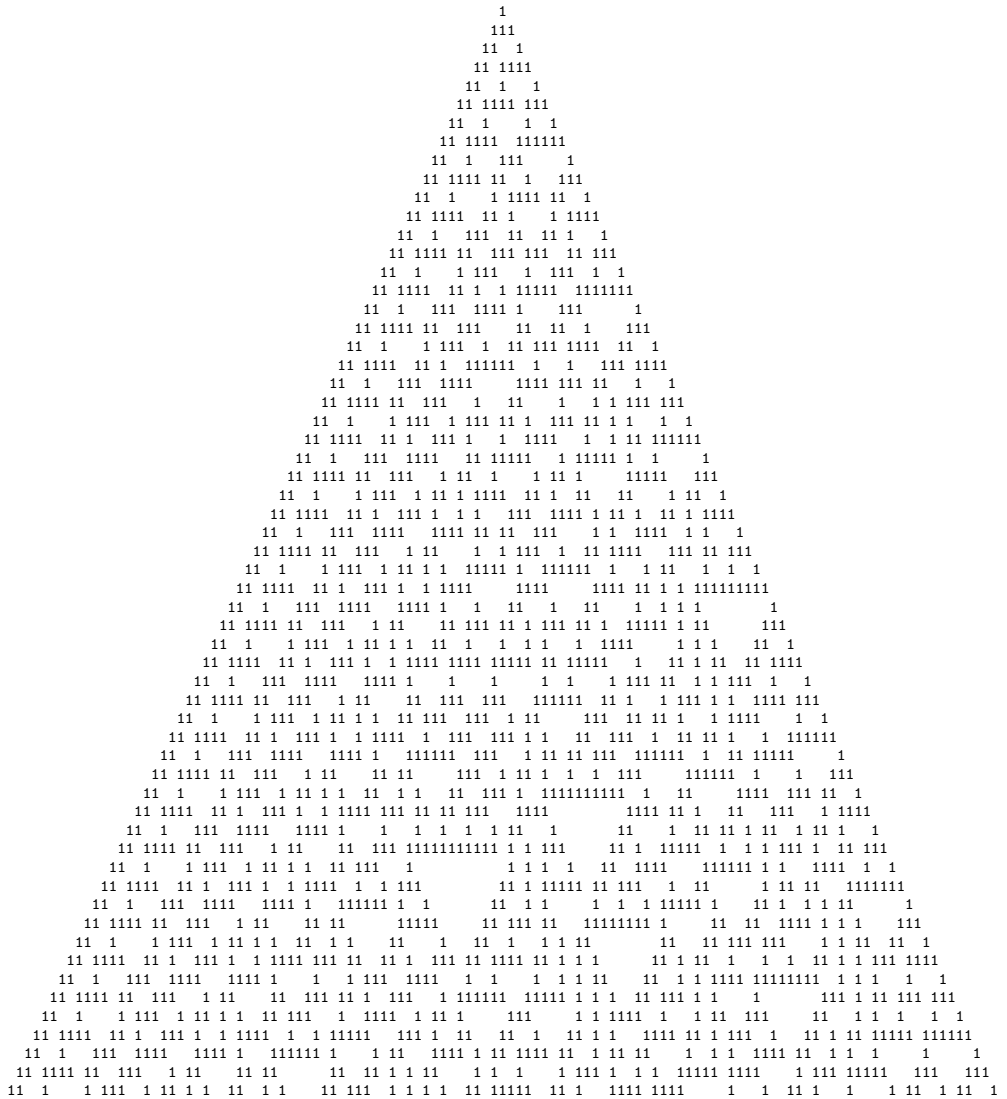


Figura 3.1: Comportarea aparent complexă a unei structuri celulare simple.

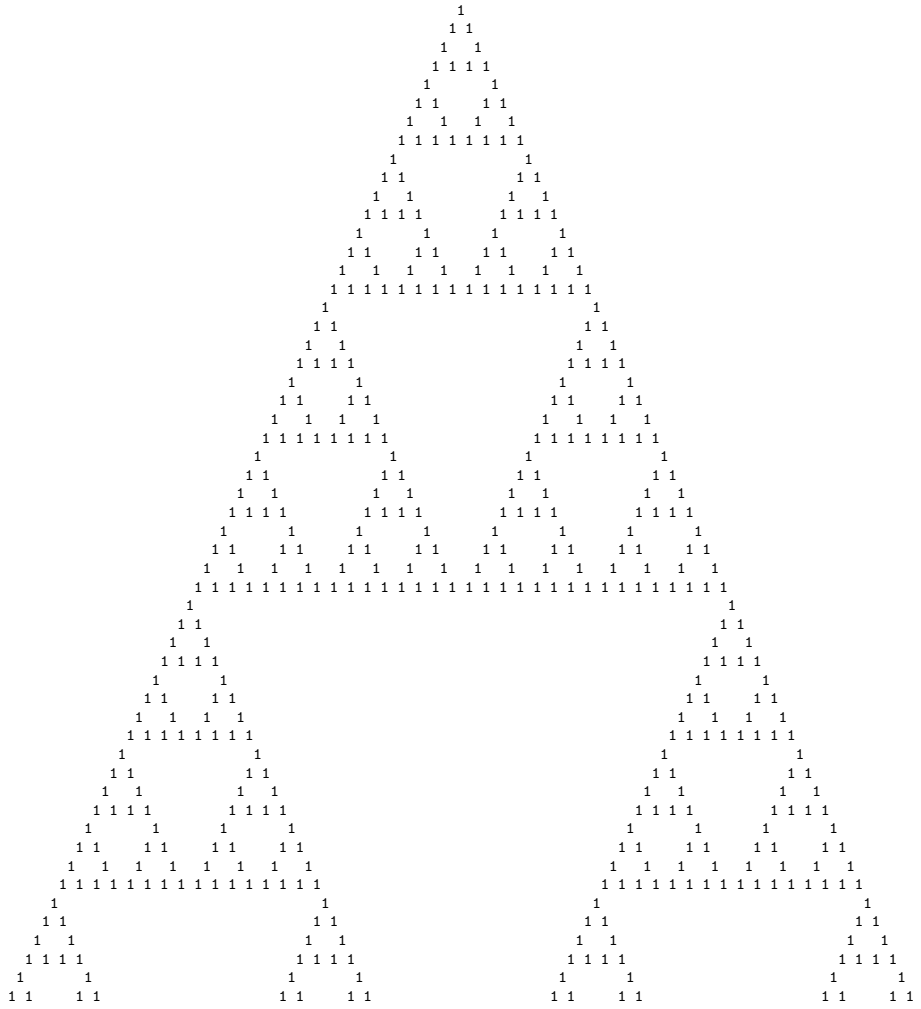


Figura 3.2: Triunghiul lui Sierpinski gnerat de un automat celular linear.

```
   input init , clock);

   always @(posedge clock)
    if (init) out <= in            ;
     else     out[n-2:1] <= out[n-3:0] ^ out[n-1:2];
endmodule
```

Funcţia

$$(A + B) \oplus C$$

implicată în fiecare celulă, în exemplul anterior, a fost substituită cu

$$B \oplus C$$

Rezultă configuraţia de biţi din Figura 3.2. Este triunghiul lui Waclaw Sierpinski, matematician polonez, care a propus-o în 1915. Configuraţia de biţi obţinută nu mai este aleatoare, dar este de o complexitate greu justificabilă prin codul simplu care a generat-o.

◇

Diferenţa cantitativă dintre cele două descrieri anterioare în Verilog nu-şi poate găsi justificarea în diferenţa de complexitate aparentă a configuraţiilor generate. Complexitatea reală a celor două configuraţii – dimensiunea definiţiilor – este aproape identică, pe când aparenţele sunt semnificativ distincte.

Triunghiul lui Sierpinski conţine un pattern, dar este vorba despre un pattern cu o evoluţie dinamică, greu de surprins chiar într-o exprimare într-un limbaj natural. Se pare, însă, că este vorba despre un proces de o simplitate profundă care, în virtutea expereinţelor curente, nu-şi poate găsi o descriere suficient de compactă. Este, din acest motiv, spectaculoasă capacitatea definiţiei date de a exprima cu o concizie foarte avansată realitatea triunghiului lui Sierpinski. Spectaculozitatea reprezentării obţinute se bazează pe un artificiu formal "adânc" pe care anumite formalismele-l permit.

Atunci când între complexitate şi aparenţă relaţia nu poate fi justificată, cu siguranţă operăm cu complexităţi aparente. Este vorba de o formă de *ordine ascunsă* pe care creatorul o controlează creând aparenţa unei complexităţi prin care-şi captează publicul. Este vorba de o complexitate controlată în laboratorul creaţiei, departe de posibilităţile de percepţie ale publicului. Este bucătăria ascunsă unde, într-o ascundere creatoare, se aplică reţete secrete prin care este generat miracolul formelor artistice.

Miracolul creaţiei artistice îşi găseşte parţial explicatia în existenţa şi fructificarea complexităţii aparente. Această fructificare este făcută mai mult sau mai puţin conştient. Despre Johann Sebastian Bach s-a spus că făcea matematică fără să o ştie. Demersul nostru este unul prin care încercăm fructuficarea conştientă a diverselor forme de complexite aparentă. Una dintre ele este cea a lumii fractalilor.

3.2 Exemple de fractali

În decaniul 8 al secolului trecut, Benoît B. Mandelbrot introduce conceptul de fractal [27]. Fractalii sunt, de regulă, construcţii realizate prin autosimilaritate. Oricât am intra în detalile construcţiei lor, vom găsi acelaşi pattern generator. Multe fenomene reale pot fi reprezente, aproximativ, prin modele fractale. Atât de multe, încât unii cred în posibilitatea descrierii fractale dincolo de limite rezonabile. Folosit cu moderaţie, modelul fractal se poate dovedi o unealtă eficientă pentru manipularea complexităţilor aparente. În spaţiul creaţiei artistice, oferă o potrivită ascundere pentru "reţete de succes".

Vom exemplifica conceptul de fractal prin câteva exemple semnificative ce vor fi utilizate şi în demersul pe care-l întreprindem.

Pulberea lui Cantor: se bazeză pe regula conform căreia unui segment de dreaptă i se suprima treimea mediană. Rezultă două segmente având lungimea de trei ori mai mică. Acestora li se aplică aceeaşi regulă. Ş.a.m.d. Rezultă secvenţa de linii întrerupte reprezentată în Figura 3.3.

Figura 3.3: **Pulberea lui Cantor.**

Curba lui Koch: este mai elaborată, deoarece, spre deosebire de *pulberea lui Cantor*, segmentul înlăturat este înlocuit cu alte două segmente ca în reprezentarea din Figura 3.4. În locul unei linii întrerupte, rezultă o linie frântă. Procesul continuă şi se generează o linie foarte complexă. Curba lui

Koch se desfăşoară bidimensional, astfel încât va permite modelarea unor procese caracterizate prin două variabile (spre exemplu, timp şi frecvenţă).

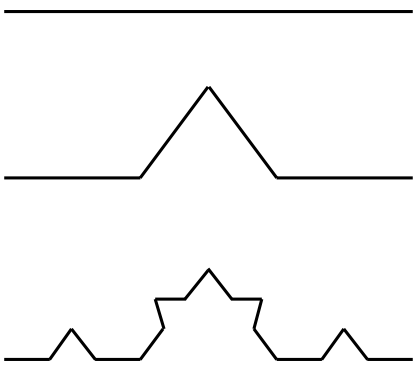


Figura 3.4: **Curba lui Koch.**

Triunghiul lui Sierpinski: este un fractal *avant la lettre*. Mandelbrot introduce conceptul de fractal în anii 1970, în timp ce Sierpinski propune o imagine construită prin auto-similaritate în 1915. Mentalul uman lucrează, de multe ori, cu anumite concepte mult înainte de a deveni conştient de posibilitatea definirii lor. În Figura 3.2 este redat triunghiul lui Sierpinski într-o formă ce permite extinderea reprezentării oricât de mult. Este foarte interesant faptul că acest fractal, poate şi alţi fractali, poate fi definit ca extinzându-se într-un spaţiu precis delimitat, sau ca extinzându-se nelimitat.

Definirea limitată porneşte de la un triunghi plin la care mijloacele laturilor sunt unite formând un triunghi similar cu vârful în jos, care este eliminat, celor trei triunghiuri rămase li se aplică acelaşi procedeu. Ş.a.m.d.

Vor rezulta două modalităţi de folosire a acestui fractal în dezvoltarea unei forme: o modalitate intensivă şi una extensivă.

Mulţimea Julia: este definită pornind de la reprezentarea famiilor de polinoame cuadratice de tipul

$$f_c(z) = z^2 + c$$

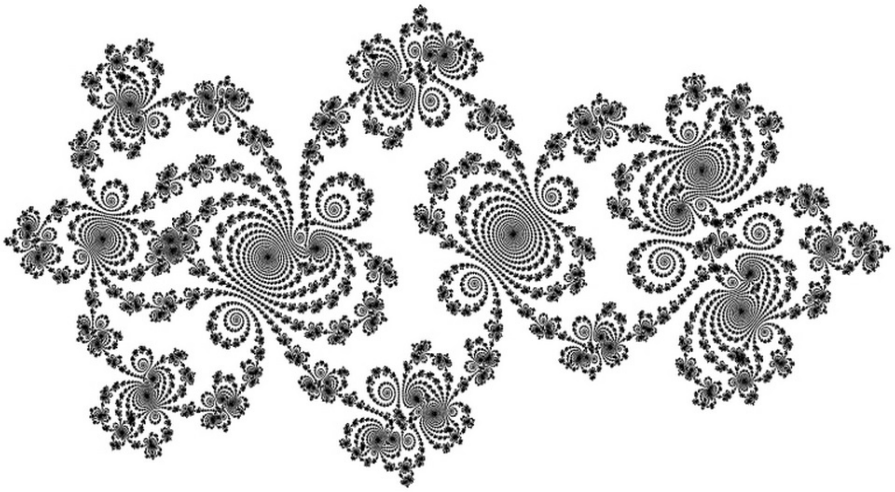


Figura 3.5: **Mulţimea Julia.**

unde c este un număr complex care influenţează foarte puternic forma concretă a fractalului. Un exemplu este dat în Figura 3.5. Nu putem să nu remarcăm simplitatea expresiei în comparaţie cu apariţia deosebit de elaborată a fractalului pe care-l generează. Oricât de elaborată ne-ar părea imaginea din Figura 3.5, va trebui să o acceptăm ca având numai o complexitate aparentă, perfect controlată de un mecanism deosebit de simplu.

Mulţimea Mandelbrot : este definită pornind de la reprezentarea familor de polinoame cuadratice complexe de tipul

$$P_c :\to z^2 + c$$

unde c este un parametru complex.

O definiţie formală compactă porneşte de la $P_c^n(z)$, care reprezintă pe $P_c(z)$ compus cu el însuşi de n ori, şi generează mulţimea Mandelbrot ca fiind submulţimea numerelor complexe definită prin:

$$M = \{c \in \mathbf{C} : \exists s \in \mathbf{R}, \forall n \in \mathbf{N}, |P_c^n(0)| \leq s\}$$

unde: $\mathbf{C}$ este mulţimea numerelor complexe, $\mathbf{R}$ este mulţimea numerelor reale, iar $\mathbf{N}$ este mulţimea numerelor întregi.

Detaliile formale anterioare nu sunt importante, în acest context, decât pentru a pune în evidenţă concizia cu care poate fi definită mulţimea nume-

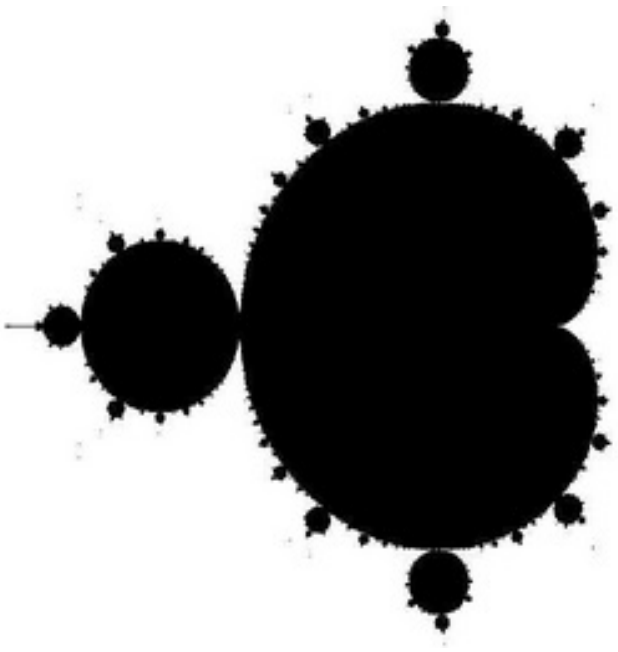


Figura 3.6: **Mulţimea Mandelbrot.**

relor din planul complex care se organizează ca în reprezentarea din Figura 3.6.

Exemplele anterioare au ilustrat caracterul aparent complex al formelor fractale. O parte dintre ele vor fi folosite în continuare pentru a ilustra demersul nostru bazat pe folosirea unor reprezentări aparent complexe pentru a construi coerent forme expresive.

3.3 Dimensiunea fractală

Conceptul de *dimensiune fractală* are o lungă evoluţie în istoria matematicii. Benoît Mandelbrot îl consacră definitiv atunci când face corelaţia dintre lungimea coastelor Marii Britanii şi unitatea de măsură cu care s-a făcut evaluarea [26]. În funcţie de unitatea de măsură folosită, vom obţine un anumit rezultat sau un alt rezultat. Dacă unitatea de măsură va fi kilometrul, atunci vom obţine o lungime totală mult mai mică decât în cazul în care unitatea de măsură va fi suta de metri (o mulţime detalii sunt eludate dacă unitatea de măsură este mare).

Dimensiunea fractală este, informal exprimat, un raport care dă socoteală asupra complexitatăţii pe care un pattern o "poartă" atunci când acesta scalează prin auto-similaritate. Dimensiunea fractală ne spune ceva despre felul în care un pattern "umple" spaţiul în care se dezvoltă. Spre exemplu, despre curba lui Koch putem spune ca "fracturează" spaţiul mult prea detaliat pentru a fi considerată unidimensională, dar nu putem spune că atinge complexitatea unei structuri bidimensionale. Ne aşteptăm şi acceptăm, în aceste condiţii, ca dimensiunea fractală a curbei lui Koch să se poziţioneze undeva între 1 şi 2.

Definiţia formală cea mai folosită pentru a caracteriza dimensiunea fractală este cea care consideră un raport între gradul detalierii şi nivelul scalării.

Definitia 3.4 *Dimensiunea fractală a unui fractal F, la care în fiecare etapă se produce un număr de părţi prin* divizarea *componentelor curente având drept consecinţă un număr de părţi* generate *ale etapei următoare, este definită prin raportul:*

$$D_F = \frac{log(numar\_de\_parti\_generate)}{log(numar\_de\_parti\_obinute\_prin\_divizare)}$$

◇

Exemplul 3.3 *Dimensiunea fractală a curbei lui Koch se calculează considerând că fiecare segment se împarte în trei părţi egale, iar noua configuraţie obţinută conţine 4 segmente. În Figura 3.4, segmentul iniţial a fost împarţit în 3 –* `numar_de_parti_obinute_prin_divizare = 3` *– iar în urma primei iteraţii am obţinut o linie "fracturată" formată din 4 segmente –* `numar_de_parti_generate = 4`*. Deci:*

$$D_{Koch} = \frac{log\,4}{log\,3} = 1.26$$

◇

Exemplul 3.4 *Dimensiunea fractală a pulberii lui Cantor, în cazul definiţiei uzuale, ce presupune înlăturarea treimii mediane a segmentului, este:*

$$D_{Cantor} = \frac{log\,2}{log\,3} = 0,63$$

◇

În geometria tradiţională, dimensiunile sunt exprimate în numere întregi. Aceste dimensiuni sunt cazuri particulare ale unor dimensiuni fractale. Dacă vom considera cazul general al unui cub cu n dimensiuni, atunci: punctul este un cub cu 0 dimensiuni, segmentul de dreaptă este un cub cu o dimensiune, pătratul este un cub cu 2 dimensiuni, iar cubul este un cub cu trei dimensiuni. În exemplul următor vom calcula dimensiunile fractale ale construcţiilor geometriei tradiţionale.

Exemplul 3.5 *Dimensiunea fractală a unui cub în spaţiul cu trei dimensiuni rezultă:*

$$D_{cub} = \frac{log\,8}{log\,2} = \frac{log\,27}{log\,3} = 3$$

pentru că, dacă laturile, L, ale cubului se împart la n se obţin 3^n *cuburi cu latura latura de* L/n.

Dimensiunea fractală a unui cub în spaţiul cu două dimensiuni rezultă:

$$D_{patrat} = \frac{log\,4}{log\,2} = \frac{log\,9}{log\,3} = 2$$

pentru că, dacă laturile, L, ale pătratului se împart la n se obţin 2^n *pătrate cu latura latura de* L/n.

Dimensiunea fractală a unui cub în spaţiul cu o dimensiune rezultă:

$$D_{segment} = \frac{log\,2}{log\,2} = 1$$

$\diamond$

Dimensiunea fractală apare ca un concept ce generalizează conceptul de dimensiune. În acelaşi timp, conceptul de fractal permite luarea în considerare a unor moduri nuanţate controlat de organizare a spaţiului. Complexitatea construcţiilor realizate în spaţiul tradiţional, cu dimensiuni măsurate prin numere întregi, depinde strict de mecanismele de structurare. În spaţiile fractale, mecanismelor de structurare complexă le putem adăuga şi proprietăţile specifice spaţiului fractal în care am optat să dezvoltăm construcţia unei forme. Complexitatea aparentă a spaţiului fractal folosit contribuie la *creştera controlată* a expresivităţii actului creator de generare şi interpretare a formelor artistice.

Instalarea "laboratorului" în intervalul dintre dimensiunile întregi, convenţionale, oferă creatorului forme de libertate suplimentare. Folosirea controlată a acestor libertăţi permite extinderea repertoriului de mijloace prin care creatorul generează miracolul unor complexităţi care transmit "înţelesuri" ce surprind în limite suportabile prin perceptibilitatea lor.

Capitolul 4

Fractali. Aplicaţii

Există, deja, o largă literatură de specialitate, privind aplicaţiile geometriei fractalilor în teoria şi practica artei muzicale. Este frapant să descoperim, de exemplu, cum Mozart anticipă (posibila) reaşezare a fractalilor pe o linie melodică. Ilustrăm aceasta printr-un fragment din aceeaşi Sonată KV 331 (vezi Figura 4.1). Este vorba de prima variaţiune a temei de la primul exemplu decupat anterior. Se vede limpede că ar fi vorba de o *sui-generis* formă a pulberii lui Cantor (vezi Figura 3.3) prezentată în Introducere. Putem considera acest exemplu ca pe o sugestie pentru folosirea ulterioară, riguroasă, a acestui fractal. Procedeul mozartian constă în folosirea unui posibil şir continuu de sunete (făcut din broderii inferioare şi superioare), din care în mod sistematic este eliminată ultima componentă a broderiei inferioare, pentru a sublinia caracterul ascendent al liniei melodice.

Pulberea lui Cantor este o structură a cărei complexitate creşte prin *eliminarea* unor secţiuni dintr-un continuu. Mai profitabilă pentru demersuri cu complexitte sporită ar fi curba lui Koch (vezi Figura 3.4). Aici, în fiecare etapă a construcţiei acesteia, zonelor eliminate li se *substituie* o structură autosimilară.

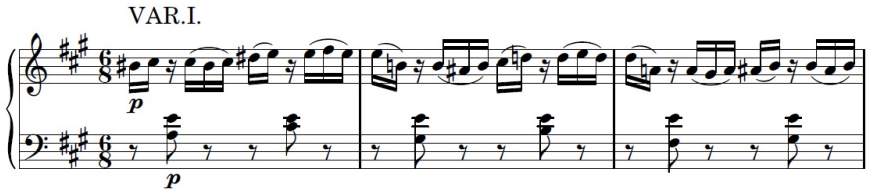


Figura 4.1: **Începutul primei variaţiuni a temei Sonatei KV 331.**

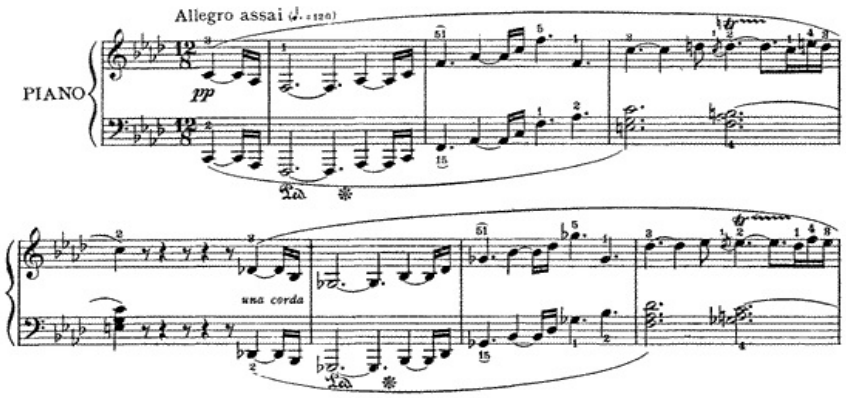


Figura 4.2: **Sonata Appassionata de Beethoven.**

În lumina acestor consideraţii (privind un fractal mai complex), o situaţie relevantă o întâlnim în muzica lui Ludwig van Beethoven, precum în celebra Sonată op. 57, în fa minor, *Appassionata* (vezi Figura 4.2). În acest caz, se poate vorbi de un procedeu mai complex prin care procesul de eliminare (prezent în pulberea lui Cantor) este imediat urmat de unul de completare (precum în curba lui Koch). Aceasta din urmă constă din modificarea unui segment prin comprimarea şi egalizarea componentelor sale (`notă-lungă - notă-scurtă - notă-lungă - notă-de-oprire` se înlocuieşte cu `notă-scurtă - notă-scurtă - notă-scurtă - notă-de-oprire`).

Există tendinţa, la unii cercetători actuali, de a asimila sugestiile, uneori foarte puternice, existente în creaţiile unor compozitori ai trecutului, unei folosiri riguroase şi conştiente a procedeelor din teoria fractalilor. În studiul de faţă se propune ca, pornind de la anticipaţiile amintite, să se treacă la folosirea riguroasă a procedeelor doar evocate metaforic în trecut.

În ambele cazuri, compozitorii au realizat, intuitiv, o aplicaţie *sui-generis* a ceea ce, mai tîrziu, va fi cuprins în teoria fractalilor, dezvoltată de B. Mandelbrot.

Un compozitor contemporan interesat de lucrul cu fractalii (mai ales cu mulţimea Julia) este György Ligeti. Şi rezultatele lui sunt, evident, de natură intuitivă, însă aproximează destul de fidel forme din domeniul fractalilor.

În pagina aleasă din lucrarea pentru orchestră mare intitulată *Atmosphères* (vezi Figura 4.3), întâlnim un număr foarte mare de microeve-

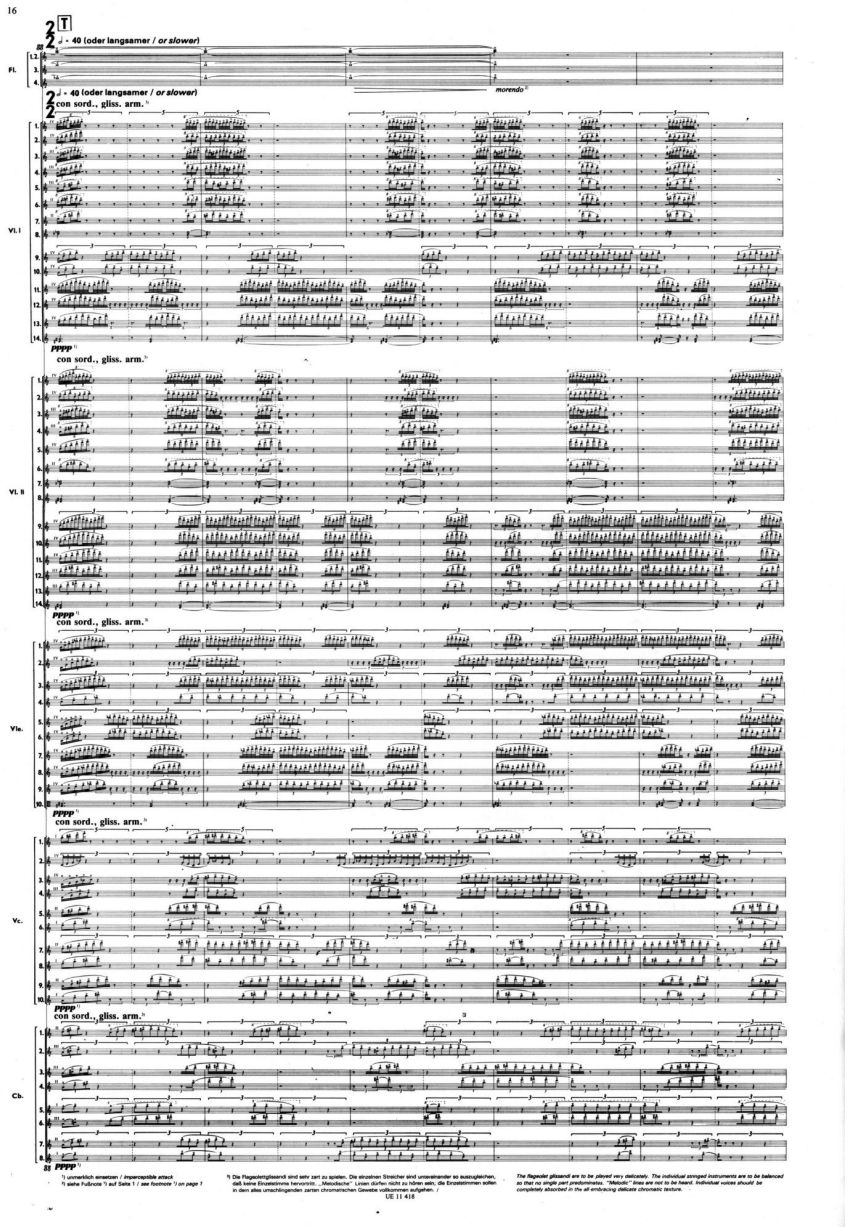


Figura 4.3: **Pagină din *Atmosphères* de György Ligeti.**

nimente, uşor diferite între ele, care se suprapun în combinaţii complicate, neregulate. Această micropolifonie trimite către microevenimentele ce se întâlnesc în fractalul Julia (vezi Figura 3.5). Structurile ce compun acest fractal sunt alcătuite din microforme realizate prin autosimilaritate, comparabile cu structurile autosimilare din partitura lui Ligeti. Compozitorul însuşi, într-un interviu, insistă asupra paralelismelor dintre muzica lui şi procedeele induse de teoria fractallilor:

> *Fractals are patterns which occur on many levels. This concept can be applied to any musical parameter. I make melodic fractals, where the pitches of a theme I dream up are used to determine a melodic shape on several levels, in space and time. I make rhythmic fractals, where a set of durations associated with a motive get stretched and compressed and maybe layered on top of each other. I make loudness fractals, where the characteristic loudness of a sound, its envelope shape, is found on several time scales. I even make fractals with the form of a piece, its instrumentation, density, range, and so on. Here I've separated the parameters of music, but in a real piece, all of these things are combined, so you might call it a fractal of fractals.*
>
> Gyorgy Ligeti, 1999 interview, The Discovery Channel

Exemplele amintite anterior au arătat în mod clar legăturile *posibile*. În cele ce urmează propunem o modalitate riguroasă – în măsura posibilului, oferită de practica muzicală – de translare a tehnicilor fractalilor în modelarea parametrilor sunetului.

4.1 Aspecte ale fractalilor în spaţiul ritmic

În cele ce urmează, se vor oferi cîteva modalităţi concrete (adică în planul scriiturii muzicale) de abordare a fractalilor în teritoriul ritmului şi al înălţimilor, adică al valorilor de note şi al frecvenţelor cu care lucrează orice muzician. Ne-am oprit la doi fractali mai simpli, respecti la *pulberea lui Cantor* şi *curba lui Koch* şi vom examina virtualităţile acestora prioritar în domeniul dar şi, în cazul curbei lui Koch, în domeniul frecvenţelor.

4.1.1 Pulberea lui Cantor

Vom începe cu cel mai simplu caz de fractalizare, respectiv, pulberea lui Cantor. Să luăm următorul exemplu. Durata ce va fi fractalizată este de 8 secunde (un sunet lung, continuu). Din punct de vedere al notaţiei muzicale, aceasta înseamnă două note întregi legate (vezi Figura 4.4), la un metronom `M.M pătrime = 60`.

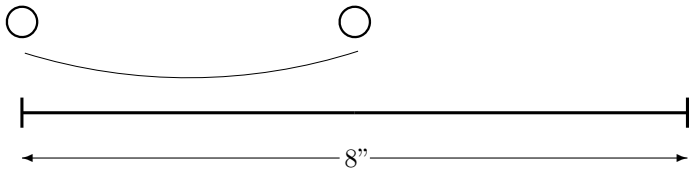


Figura 4.4: Segmentul de timp ce va fi supus fractalizării.

Vom considera că regula de fractalizare, ce acţionează în acest caz, este eliminarea unui fragment reprezentând 1/6 din lungimea segmentului, fragment plasat în centrul acestuia. Geometric, dar şi muzical, procesul se concretizează prin împărţirea în 12 secţiuni egale (vezi Figura 4.5) şi eliminarea a două secţiuni din centru (vezi Figura 4.6).

Notaţia muzicală echivalentă este:

- cele două note întregi vor fi înlocuite prin $6 + 6$ valori de triolete de pătrimi (Figura 4.5)
- ultima pătrime a celui de al doilea triolet şi prima pătrime a trioletului următor sunt înlocuite prin pauzele corespunzătoare (Figura 4.5.

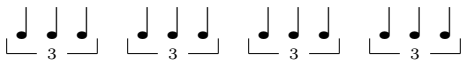


Figura 4.5: **Împărţirea segmentului de timp în 12 părti egale.**

Figura 4.6: **Reprezentarea muzicală a primei aplicaţii a pulberii lui Cantor.**

Dimensiunea fractală rezultată este:

$$\frac{log\ 2}{log\ 12}$$

Lucrurile se complică atunci când fractalizarea continuă şi trebuie extrasă valoarea 1/6 din cele două grupe rămase (de 2 ori 5 pătrimi de triolet). Să consideră o grupă de 5 pătrimi de triolet. Pentru a aplica aceeaşi regulă, trebuie să eliminăm o şesime din cele 5 pătrimi de triolet. Aceasta ar corespunde eliminării a 83% (prin rotunjire 80% din pătrimea de triolet centrală.

Dacă vom folosi notaţia muzicală acceptată, atunci rezultă, aproximativ, eliminarea unui fragment central care reprezintă $1/6,67$ în loc de $1/6$. În cazul în care am persista să găsim o modalitate de notare muzicală riguroasă, ceea ce am obţine ar deveni imposibil de executat, ca şi de perceput.

În ultimiii ani, unele dintre cercetăarile cele mai avansata si mai conceputal susţinute în domeniul complexităţii ritmului, s-au realizat la IRCAM (Institute de Recherches et Coordination Acoustique/Musique), Paris. De pildă, este de menţionat studiul întreprins de câţiva cercetători şi colaboratori ai institutuuli parizian [1].

4.1.2 Curba lui Koch

Reamintim configuraţia curbei lui Koch, reprezentată in Figura 3.4. Vom presupune că această curbă reprezintă desfăşurarea unui contur melodic. Axa timpului este linia orizontală. Pentru simplificare, presupunem că avem un singur sunet ţinut, Do (vezi Figura 4.7a). Vom urmări ce se întâmplă atunci când curba lui Koch începe să se dezvolte prin autosimilaritate. În Figura 4.7b, am obţinut, datorită folosirii unui unghi de 45^0 şi datorită faptului că vom considera proiecţia pe orizontală (respectiv, pe axa timpului) a liniei oblice, un sunet cu frecvenţa egală cu cea a lui Do înmulţită cu $\sqrt{2}$. Noua frecvenţă este, evident, Fa#.

În continuare, în Figura 4.7c, datorită aplicării regulii curbei, apar proiecţii suplimentare. Pe fiecare dintre cele patru segmente, se produc noi proiecţii. Unele dintre acestea, de pe segmentele oblice, vor genera proiecţii ale proioecţiilor. Pe lângă *Fa#*-ul, deja ştiut, multiplicarea lui $Do \times \sqrt{2}$, pe segmentele oblice, cu încă un $\sqrt{2}$ aduce încă un Do, octava celui precedent. Continuând, în Figura 4.7d, apar încă un Fa# şi Do. Însumând, spre exemplu, sunetele obţinute pornind de la un Do prin aplicarea de 2 ori a algoritmului curbei lui Koch, obţinem următoarea succesiune de sunete:

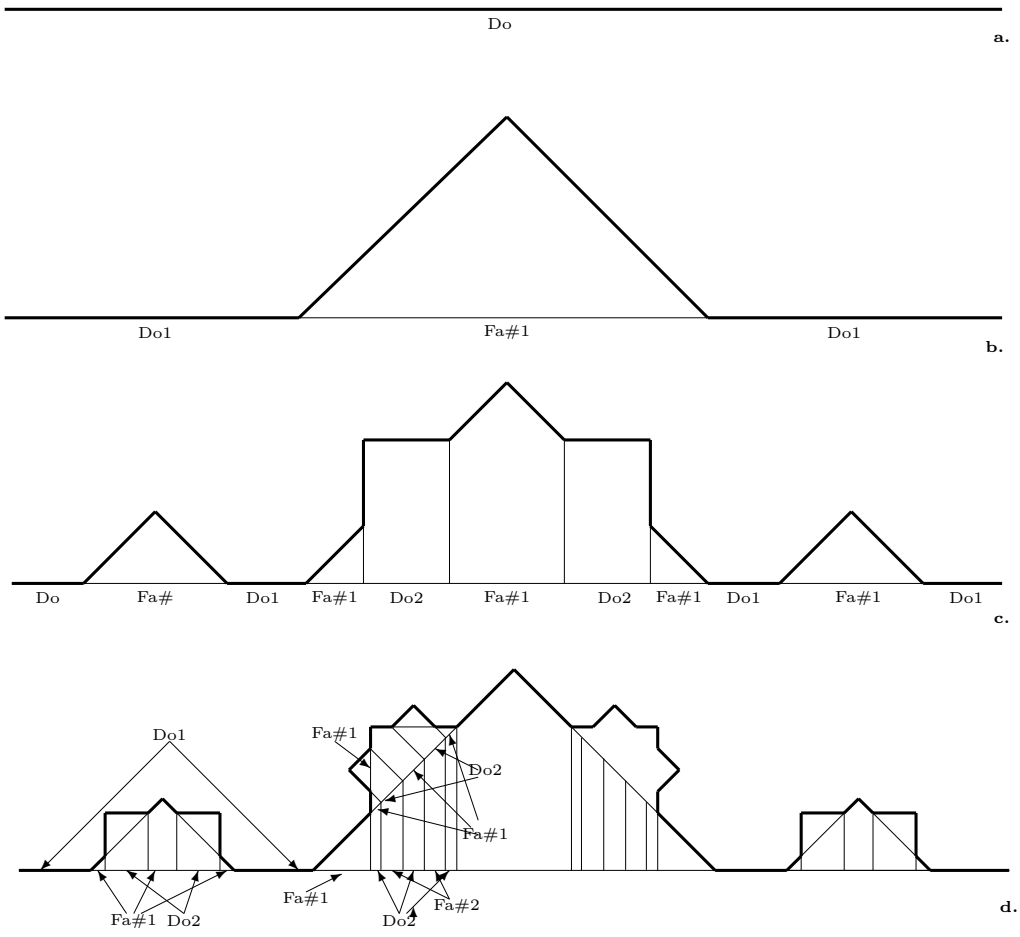


Figura 4.7: "Fracturarea" unui Do pe curba lui Koch.

Do, Fa#, Do, Fa#, Do1, Fa#, Do1, Fa#, Do1, Fa#, Do1

În cazul celei de treia aplicaţii, apărând un al treilea Do, lucrurile sunt încă şi mai complicate.

Din cele expuse până acum, rezultă evident că alegerea unui unghi mai mic de 45^0, prezintă situaţii mult mai "interesante" din punct de vedere sonor, ieşindu-se, eventual, din sistemul temperat.

Curba lui Koch, prin alegerea pe cam făcut-o (proiecţia continuă pe axa locală) realizează şi o comprimare în timp.

Procedeul tocmai exemplificat este aplicabil unei secvenţe sonore de tip melodic, ceea ce însemnă că în loc de un singur sunet vor exista mai multe.

Pentru eficienţa sporită a utilizării acestui procedeu, este de dorit să se realizeze un program care să permită utilizarea unui spectru larg de unghiuri de proiecţie, rezultatele aflându-se, astfel, la dispoziţia compozitorului.

Pentru viitor, ar fi important de studiat extensia procedeelor fractale asupra parametrilor sonori mai greu cuantificabili, respectiv dinamica şi timbrul.

Capitolul 5

Transformări şi forme

Muzica este arta cea mai direct legată de scurgerea ireversibilă a timpului. Aceasta înseamnă că procesele pe care le subsumăm idei de formă muzicală desfăşurându-se pe această axă a timpului vor avea un număr de proprietăţi ce ţin de această particularitate. Altfel spus, forma muzicală este o procesualitate al cărei *output* este diferit de *input* (deci, parametrii iniţiali ai procesului se vor regăsi modificaţi ireversibil în final).

Un exemplu celebru, în care premoniţia autorului este cu totul remarcabilă, este *Arietta con variazioni*, finalul ultimei sonate, Op. 111 în Do minor, pentru pian de Beethoven. Tema este foarte simplă, ea suferă un număr foarte mare de transformări, iar la sfârşit asistăm la disoluţia parametrilor ei în întregul spaţiu sonor al pianului (faceţi click aici: `http://www.youtube.com/watch?v=czoO25_J4I8`).

Putem vorbi despre o premoniţie muzicală asupra unor descoperiri care vor fi făcute mai târziu de către Saadi Carnot, Ludwig Eduard Boltzmann şi Ilya Prigogine [31] [32]. Primul dintre aceştia, francezul Carnot, este creatorul unui foarte controversat, la vremea lui, principiu fizic numit *al doilea principiu la termodinamicii*, conform căruia evoluţia naturală a unui sistem fizic închis, care nu realizează schimburi cu mediul încojurător, se îndreaptă către o stare de echilibru caracrerizată prin ordine minimă (acest lucru se obţine prin distrucţia oricărei structurări interne).

Mai târziu, către sfârşitul secolului al XIX-lea, austriacul Boltzmann a formulat mai riguros şi a extins acest principiu, folosind statistica matematică.

În sfârşit, în a doua jumătate a secolului al XX-lea, laureatul din 1977 al Premiului Nobel pentru chimie, Ilya Prigogine, reconsideră aproape toate

domeniile activităţii omeneşti de la ştiinţă până la societate trecând prin diverse forme de manifestare culturală. Tocmai din acest motiv, cercetările lui Prigogine au o relevanţă deosebită pentru gândirea muzicală.

În acest capitol, va fi vorba despre o strategie a organizării formei muzicale ca procesualitate pe axa timpului. Organizarea propusă ţine de două *procese de emergenţă*. Pe scurt, este vorba, mai întâi, de un proces iterativ, prin care, pornind de la un organism sonor dat, se desfăşoară un şir de *transformări* folosite în cel de al doilea proces, cel de *emergenţă a formei muzicale*.

5.1 Naşterea formei sonore

Naştera formei sonore este o problemă esenţială în proiectul de faţă. Ea este o realitate bazată genetic pe conceptul de organism sonor. Propunem următoarele două definiţii.

Definitia 5.1 *Organismul sonor este o asamblare de microevenimente formate din diferite combinaţii ale parametrilor sunetului.*

◇

Definitia 5.2 *Forma sonoră este o asamblare de organisme sonore care întreţin relaţii de transformare prin conexiune directă.*

◇

În Figura 5.1 sunt prezentate modalităţile de creştere a numărului de organisme sonore şi de organizare a înlănţuirilor lor. Acestea din urmă pot fi de tip *serie* (Figura 5.1a), *paralel* (Figura 5.1b) şi *serie-paralel* (Figura 5.1c). În matematică există un procedeu de formalizare a ceea ce tocmai am prezentat. Este vorba despre regula de compoziţie a funcţiilor, exprimată prin expresia:

$$f(x) = g(h_1(x), h_2(x), \ldots h_n(x))$$

Dacă $n = 1$, atunci avem de a face cu o extensie funcţională de tip serie: funcţia f este calculată prin înserierea operaţiilor presupuse de funcţiile h şi g.

Dacă funcţia g este de tip identitate, atunci funcţia f se reduce la calculul în paralel a n funcţii de tip h_i.

În Figura 5.1d se arată cum este posibilă şi o succesiune nerecomandabilă din punctul de vedere al impactului informaţional. Recte, o transformare

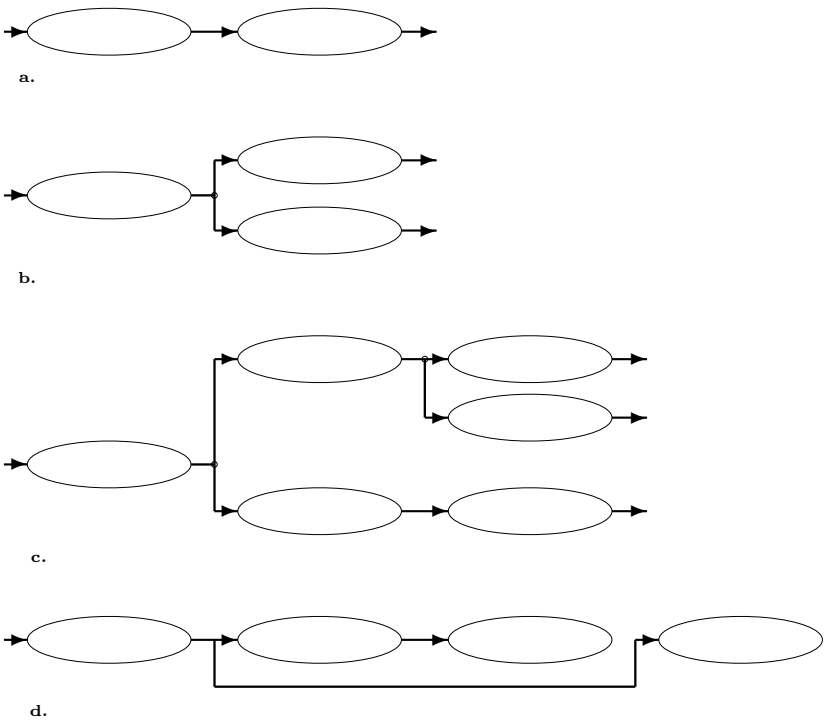


Figura 5.1: **Tipuri de organizare în timp a fromei sonore.**

provenită din organismul iniţial apare, întârziat, după ce alte transformări mai avansate şi-au consumat deja impactul.

Din Figura 5.1 observăm cum, dintr-o multitudine de organisme sonore, generate dintr-un prim organism prin procedee transformaţionale, pot emerge forme sonore complexe.

Aşa cum a observat cu sagacitate compozitorul francez Fabien Lévy [25], problema generării formei muzicale este, direct, o problemă a ajungeriii la limitele scriiturii tradiţionale occidentale; această scriitură a fos concepută pentru un anume tip de gândire sonoră care a prezidat câteva secole. Implicit, chestiiunile legate de un nou concept al devenirii sonore sunt legate de dilema păstrării (măcr parţiale) sau abandonării totale a acestei scriituri "istorice".

Identificăm, în continuare, două tipuri de transformări: trasformări discrete şi transformări continue.

5.2 Transformări discrete

5.2.1 Organismul sonor

Punctul de pornire al demersului nostru este un organism generat prin procese descrise în secţiunile anterioare. Vom începe, deci, de la o structură închisă, în sensul izolării unui sistem în cazul principiului lui Carnot.

Pentru cei familiarizaţi cu limbajul muzicologiei trediţionale, microevenimentul poate fi asimilat, aproximativ, noţiunii de *celulă muzicală*.

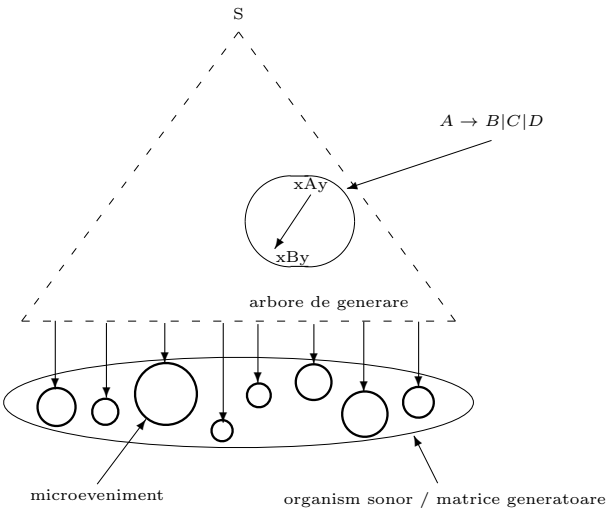


Figura 5.2: Generarea unui organism sonor.

În viziunea adoptată în studiul prezent, organismul sonor este generat (de aici şi ideea de organism) de o gramatică generativă, G, definită în cadrul unui sistem compoziţional, controlat de coordonatele unui proiect. În Figura 5.2, este prezentată existenţa unui organism sonor şi felul în care a fost generat. Elipsa, care este anvelopa virtuală a acestui organism sonor, conţine o serie de microevenimente, figurate prin cerculeţe. Microevenimentele sunt produsul unei aplicări a gramaticii G pornind de la simbolul de start, S. Arborele de generare a fost simbolizat printr-un triunghi, în care am explicitat cum în nodul xAy se aplică una din regulile $A \to B|C|D$, rezultând xBy. La

nivelul propriu-zis al organismului sonor, cele arătate se concretizează prin microevenimentele către care se îndreaptă săgeţile finale ale arborelui de generare.

Transformarea organismului sonor

În Figura 5.3 se prezintă modul în care dintr-un organism sonor iniţial se obţine o transformare pe care o vom numi *transformare genetică*.

Continuăm cu detalierea procesului descris schematic în Figura 5.3. Transforamrea presupune că a doua înfăţişare a organismului sonor suferă, la nivelul unor microevenimente, nişte distorsiuni (care în acest caz funcţionează ca o conotaţie a conceptului de modificare, prin urmare o conotaţie pozitivă). Dirstorsiunile se datorează fapului că, în anumite noduri ale grafului de generare, opţiunile s-au schimbat. În loc de $A \to B$ în nodul xAy s-a optat pentru varianta $A \to D$. Automat, microevenimentele incidentate se schimbă.

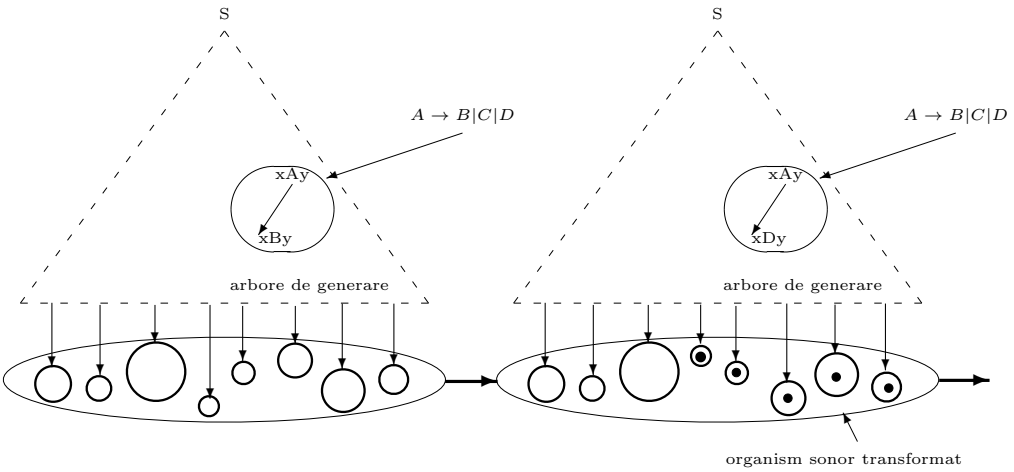


Figura 5.3: Transformarea unui organism sonor.

5.3 Transformări continue

Aceste transformări, pe care le-am denumit continue, pornesc de la un dat experimental pe care l-au observat mai mulţi muzicieni care au lucrat cu mijloace electronice de tip analogic, în speţă, înregistrările pe benzi de magnetofon). Unul dintre aceşti muzicieni este marele compozitor german Karlheinz Stockhausen [36], care a intuit primul virtuţile acestui procedeu pentru conceperea unui tip de sintaxă sonoră inedită. Mai recent, compozitorul francez Tristan Murail a aplicat simularea procedeului în compoziţia sa intitulată *Mémoire/Erosion*. În general, procedeul este cunoscut sub titulatura de *buclă de reinjecţie* (*re-injection loop*). În textul de faţă, vom mai vorbi despre acest lucru în cele ce urmează.

Metafora benzii de magnetofon porneşte de la faptul că suportul material, respectiv banda de magnetofon şi aparatul de înregistrare au un număr de imperfecţiuni "genetice". O bandă de frecvenţe limitată, zgomot de fond, neliniarităţi, defecţiuni mecanice (întinderea benzii, spre exemplu), şi altele. Există un efect cumulativ al acestor imperfecţiuni care se manifestă prin degradarea înregistrării. Această degradare se reduce, în principal, la două efecte: o filtrare a componentelor spectrale şi adăugarea unui semnal de zgomot.

Această metaforă poate fi pusă în paralel cu o realitate mai abstractă care este desfăşurarea în timp a ceea numim forma muzicală. Aceasta din urmă este o procesualitate care implică, în natura ei intimă, "degradarea" în parcurgerea axei timpului. Existenţa acestui paralelism poate sugera o modalitate coerentă şi organizată de generare a formei muzicale prin transformări continue ale unei entităţi iniţiale.

Din punct de vedere funcţional, transformările continue presupun două tipuri de acţiuni:

- o funcţie de filtrare-amplificare
- o funcţie de compunere cu un semnal ce rezultă din "tasarea" componentelor spectru din semnal.

5.3.1 Filtrare/amplificare

Pornim de la definirea unui circuit care se comportă ca un amplificator cu o caracteristică de frecvenţă elaborată. Definim astfel dispozitivul de filtrare/amplificare. În Figura 5.4 am reprezentat, în ordine, caracteristica de

transfer a filtrului amplificator, spectrul de armonice al semnalului de intrare şi spectrul de armonice rezultat din procesul de fitrare amplificare.

Spre deosebire de un filtru obişnuit, care se comportă doar ca un atenuator, filtrul amplificator realizează, simultan, o atenuare dependentă de frecvenţă şi o amplificare. Amplificarea permite, ca pe lângă atenuarea amintită, să se poată produce şi amplificarea anumitor componente din spectru.

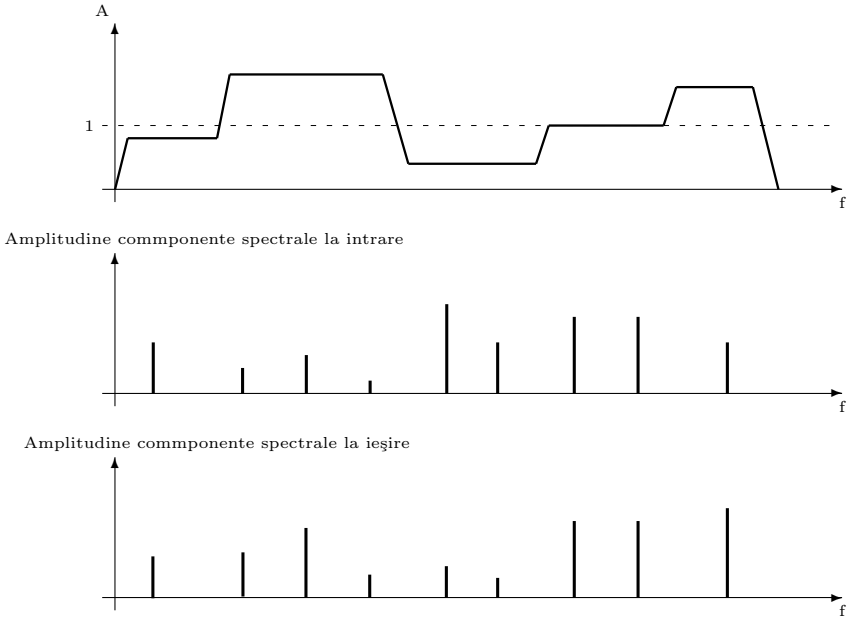


Figura 5.4: Filtrare/amplificare.

5.3.2 Zgomot prin "tasare"

Zgomotul ce se adaugă va fi aproximat cu ajutorul unui semnal obţinut prin ceea ce se poate numi "tasarea" componentelor spectrale, adică apropierea acestora şi gruparea lor într-un interval pe care l-am definit, spre exemplu, ca fiind cel dintre cea mai de jos componentă spectrală şi cea imediat următoare. Aceasta este una dintre multiplele posibilităţi de realizare a procedeului de "tasare". Dar în orice caz trebuie ca întevalul în care se regăsesc componen-

tele după "tasare" să fie mai mic, în Hertzi, decât frecvenţa celei mai joase fundamentale.

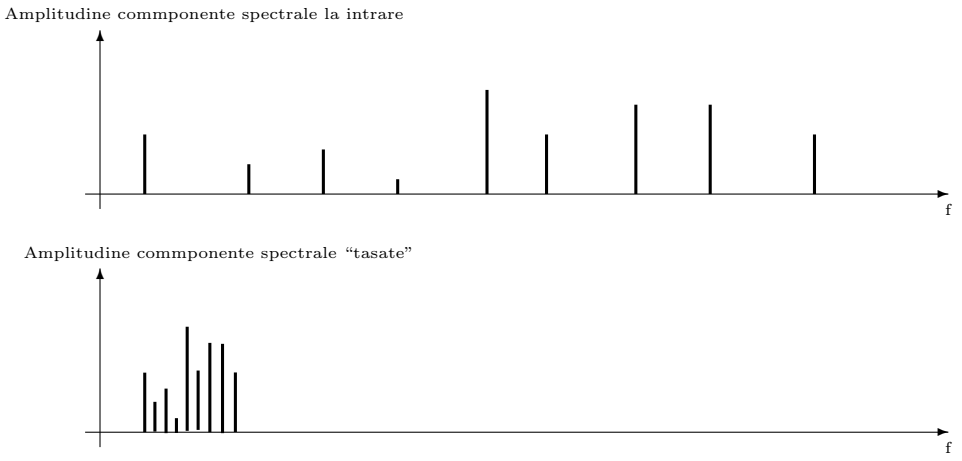


Figura 5.5: "Tasare".

Spre edificare, dăm următoarea situaţie: avem o fundamentală ($f^1 = 100Hz$) şi armonicile ei 2, 3, 5, 7 ($f^2 = 200Hz$, $f^3 = 300Hz$, $f^5 = 500Hz$, $f^7 = 700Hz$). Prin "tasare", le aducem pe toate în intervalul $\Delta f = (100Hz, 200Hz)$. Pentru aceasta vom aplica următoarea formulă:

$$f_T^i = f^i - \alpha \times \Delta f \times (i - 1)$$

pentru $i = 2, 3, 5, 7$, şi vom obţine frecvenţele "tasate",f_T^i. Pentru $\alpha = 0,9$ va rezulta următorul spectru "tasat":

$$\{100Hz, 110Hz, 120Hz, 140Hz, 160Hz\}$$

Gradul de compresie α se determină, în funcţie de numărul de componente ale spectrului de "tasat", cu formula:

$$\alpha = 1 - \frac{1}{n}$$

unde n este numărul de componente ale spectrului de "tasat".

În Figura 5.5 se prezintă spectrul iniţial şi, în a doua spectrogramă, rezultatul "tasării".

5.3.3 FAZT: sistemul fizic de transformare

În mod practic, cele două funcţii, anterior descrise, se realizează cu ajutorul arhitecturii prezentate în Figura 5.6, unde:

Fiftru/Amplificator: realizează funcţia descrisă în paragraful 5.3.1

Generator de zgomot prin "tasare": realizează funcţia descrisă în paragraful 5.3.2

iar rezultatul celor două operaţii este însumat la ieşire.

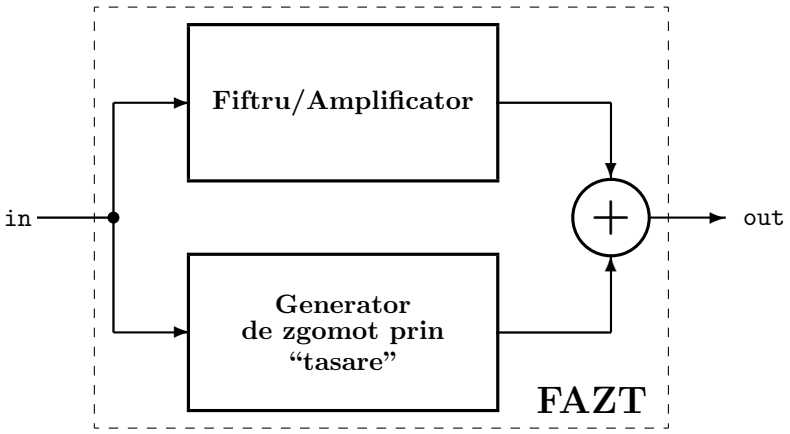


Figura 5.6: FAZT: **F**iltru **A**mplificator şi generator de **Z**gomot prin **T**asare.

Această arhitectură se poate implementa în două feluri: (1) sub forma unui program pe un calculator suficient de puternic sau (2) sub forma unui dispozitiv electronic.

Reamintim că, aplicând aceaste proceduri, obţinem invazia treptată a "zgomotului" şi distorsiunea anvelopei frecvenţiale la nivelul organismului sonor iniţial. Astfel, rezultă ceea ce ne-am propus, o formă muzicală prin transformări continue.

Partea II

Live electronics

Capitolul 6

Live electronics

În secţiunile anterioare ne-am ocupat cu probleme de principiu, teoretice, ale realizării unor partituri muzicale. Orice partitură, însă, trebuie convertită în semnal sonor prin procesul de interpretare. Cu alte cuvinte, structura ideal-formală a partiturii se transformă în fenomenologie sonoră. Această transformare se realizează cu ajutorul instrumentelor acustice mânuite de instrumentişti; cu toate restricţiile pe care le implică limitele posibilităţilor şi educaţiei umane. Din acest motiv, a apărut şi s-a dezvoltat tehnica transformării în timp real a semnalului sonor. Dar vom folosi mai departe sintagma, consacrată în limba engleză, ***live electronics***. În cele ce urmează, vom arăta un număr de procedee de *live electronics* cu menţiunea de a fi fost gândite cât mai riguros adaptate la logica construcţiei partiturii muzicale. Evităm, astfel, viziunea impresionistă a bogăţiei de procedee ce există actualmente în domeniul *live electronics*. Cu alte cuvinte, propunem un instrument fidel adaptabil logicii intrinsec muzicale ale cărei premise le-am elaborat în capitolele anterioare.

6.1 Semnalul sonor

Pentru înţelegerea şi pentru lucrul în spaţiul transformărilor în timp real (*live electronics*) ale sunetului, este necesar să ne reamintim un număr de cunoştinţe din acustică şi electroacustică.

Un sunet este o suprapunere de semnale sinusoidale organizate în trei categorii:

- fundamentala: semnalul sinusoidal cu frecvanţa cea mai joasă

$$x_1 = a_1 sin 2\pi(Nt + \phi_1)$$

- mulţimea semnalelor *parţiale* care sunt sinusoide cu frecvenţe multipli ai frecvenţei fundamentale:

$$x_1 = a_1 sin 2\pi(n_1 t + \phi_1)$$
$$x_2 = a_2 sin 2\pi(n_2 t + \phi_2)$$
$$\dots$$
$$x_p = a_p sin 2\pi(n_p t + \phi_p)$$

în care se poate evidenţia submulţimea *armonicelor*: parţialele obţinute prin multiplicare, cu un întreg, a frecvenţei fundamentale; această mulţime este ordonată, dupa cum urmează:

$$x_1 = a_1 sin 2\pi(2Nt + \phi_1)$$
$$x_2 = a_2 sin 2\pi(3Nt + \phi_2)$$
$$\dots$$
$$x_p = a_p sin 2\pi(pNt + \phi_p)$$

Parţialele care nu sunt armonice se numesc *inarmonice*.

Exemplul 6.1 *Fie semnalul:*

$$x = 5sin2\pi(66t + \phi_1) + 2sin2\pi(132t + \phi_2) + 3sin2\pi(198t + \phi_3) +$$
$$4sin2\pi(100t + \phi_4) + 1.7sin2\pi(124t + \phi_5)$$

unde avem: fundamentala (armonicul 1), un semnal cu frevenţa de 66 Hz, urmată de armonicul al 2-lea şi al 3-lea şi încheiat cu două inarmonice. ◇

Dacă în cazul "construcţiei" unui semnal putem considera în egală măsură armonice şi inarmonice, ca în exemplul anterior, în descompunerea unui semnal real nu putem lua in considerare decât semnalele armonice. Instrumentul

formal care ne peremite identificarea compoziţiei spectrale a unui semnal este *seria Fourier*:

$$x = f(t) = a_0 + a_1 sin2\pi(Nt + \phi_1) + a_2 sin2\pi(2Nt + \phi_1) +$$
$$a_3 sin2\pi(n_2 t + \phi_2) + \ldots + a_p sin2\pi(pNt + \phi_p)$$

care se tranformă în:

$$f(t) = A_1 sin2\pi Nt + A_2 sin4\pi Nt + \ldots + A_p sin2\pi pNt + \ldots +$$
$$B_0 + B_1 cos2\pi Nt + B_2 cos4\pi Nt + \ldots + B_p cos2\pi pNt$$

unde coeficienţii sunt calculaţi cu următoarele formule:

$$A_i = 2N \int_0^T f(t) sin2\pi pNtdt$$

$$B_i = 2N \int_0^T f(t) cos2\pi pNtdt$$

obţinându-se astfel transformarea informaţiei de amplitudine şi fază – a_i, ϕ_i – în amplitudine – A_i, B_i.

6.2 Spectre

Spectrele unui sunet pot fi de două feluri:

- cu conţinut exclusiv de armonice, numite *spectre armonice*
- cu conţinut de armonice şi inarmonice, numite *spectre inarmonice*.

În celebra sa lucrare *The Synthesis of Complex Audio Spectra by Means of Frequency Modulation*, John Chowning [16] (vezi şi [17]) propune următoarea formulă (anterior evocată) pentru calcularea componentelor inarmonice ale unui sunet:

$$f_n = f_0 \pm n \times f_m$$

unde f_0 este frecvenţa modulată, f_m frecvenţa modulantă, iar n intreg pozitiv. Frecvenţă f_n este rezultanta aplicării - cu plus sau minus - de n ori, a lui f_m peste f_0. Din această formulă, se vede imediat, că dacă $f_m < f_0$ în mod substanţial, se obţine o serie de f_n extrem de "aspră" la ureche (foarte nearmonică). UNul dintre compozitorii care au aplicat în mod sistematic acest procedeu în creaţia lor este francezul Gérard Grisey (ciclul de compoziţii intitulat *Les espaces acoustiques*).

Exemplul 6.2 *Fie:* $f_0 = 600Hz$ *şi* $f_m = 50Hz$. *Vor rezulta următoarele componente, deasupra lui* $600Hz$*:*
$600Hz + 50Hz$,
$600Hz + 100Hz$
...
sau dedesubtul lui $600Hz$*:*
$600Hz - 50Hz$,
$600Hz - 100Hz$
....
◇

Din exemplul anterior, este evident faptul că spectrele de inarmonice sunt alcătuite din frecvenţe foarte îndepărtate de şirul multiplilor fundamentalei având valoarea numerelor întregi.

6.2.1 Modificări de spectre

Pornind de la *formula canonică* a unui spectru de armonice

$$f_n = n \times f_0$$

putem concepe posibile transformări (modificări) ale acestuia.

Prima, şi cea mai simplă, este translatarea, mai sus sau mai jos, a spectrului (frecvenţei fundamentale i se adaugă sau i se scade o frecvenţă). Rezultă o *serie armonică translatată*:

$$f_m = (n \pm k) \times f_0$$

Următoarea este:

$$f_n = n^{\alpha} f_0$$

unde $\alpha > 1$, reprezentând o *serie armonică dilatată* (expandată).

O altă transformare este:

$$f_n = n^{\beta} f_0$$

unde $\beta < 0$, reprezentând, astfel, o *serie armonică comprimată*.

O altă modalitate, foarte simplă, de generare a unui spectru cu componente inarmonice este aceasta: oricare armonic cu un număr impar (3,5,7,...) este împărţit la una din puterile lui 2.

$$f_i = f_0 \frac{2n-1}{2^m}$$

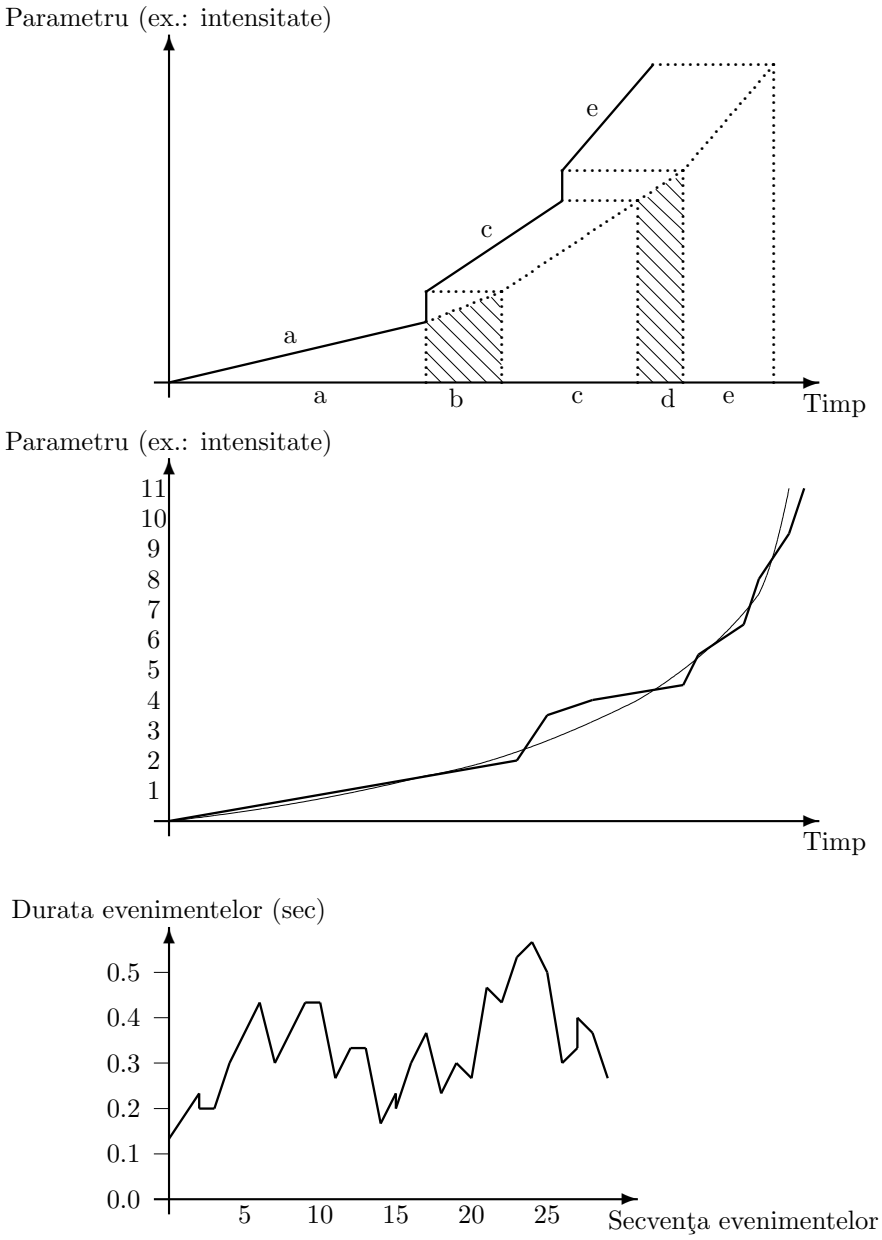


Figura 6.1: **Serii de durate [29].** **a.** Serie obţinută prin eliziunea unor intervale de timp. **b.** Serie obţinută prin abatere statistică. **c.** Zgomot de durate.

Rezultatul va fi, automat, un multiplu neîntreg al fundamentalei, deci, un inarmonic. Lucrul acesta ne arată că, cu cât o componentă a spectrului de armonice (cu indice impar) se apropie de fundamentală, efectul asupra întregului spectru este mai "aspru".

Poate fi concepută, de asemenea, o serie "armonică" de durate, după diverse procedee care imită situaţii din spectrele reale de armonice şi inarmonice (vezi Figura 6.1). În figura 6.1a, tranziţiile lente dintre procesele a şi c, respectiv între procesele c şi e au fost eliminate, cu rezultatul apariţiei unor salturi bruşte în timp. Să presuspunem, pentru mai multă claritate, că parametrul de pe ordonată ar fi intensitatea. În acest caz, creşterea constantă şi lentă a intensităţii este întreruptă de două ori prin mutarea bruscă la nivele superioare. În figura 6.1b, întâlnim un proces de accelerare (care poate fi eventual şi al intensităţii) care este perturbat stochastic. În figura 6.1c, întâlnim un caz de dezordine ritmică corespondent al zgomotului (alb ori roz) din domeniul frecvenţelor.

6.2.2 Formanţi

Atunci când spectrele anterior construite apar ca emanaţie a unor fundamentle executate punctual de un anumit instrument sau anumită voce, se manifestă şi pregnanţa anumitor zone ale acestor spectre, definitorii pentru "culoarea" lor, numite ***zone formantice***. Pentru edificare, reproducem mai jos tabloul conţinând zonele formantice ale celor 5 vocale principale.

```
Vowel   Main formant region
u          200 - 400 Hz
o          400 - 600 Hz
a          800 - 1200 Hz
e   400 - 600 and 2200 - 2600 Hz
i   200 - 400 and 3000 - 3500 Hz
```

În general, nu se poate vorbi de formanţii consoanelor pentru că acestea sunt întotdeuna anexate vocalelor şi sunt fenomene acustice tranzitorii. Prin analogie cu vocalele din emisia sonoră umană, se poate vorbi şi de zonele formantice ale instrumentelor. Există diverse procedee de manipulare a acestor zone formantice. De exemplu, la instrumentele de suflat din alamă, setul de surdine este folosit pentru construirea, prin filtrare, a zonelor formantice dorite.

Calculatorul poate fi folosit pentru construirea unor zone formantice virtuale ce pot fi aranjate astfel încât să fie obţinute timbre vocale sau/şi instrumentale inedite.

6.3 Anvelope sonore

Important pentru cele ce urmează este să ştim cîteva lucruri despre diferitele anvelope ce făuresc amprenta unui sunet.

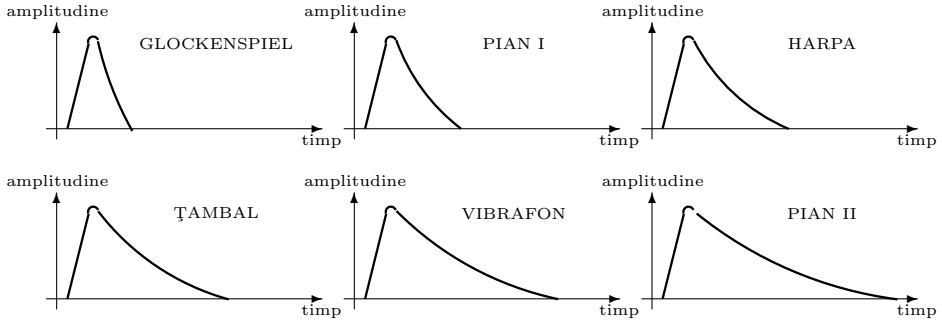


Figura 6.2: **Modele de anvelope dinamice ale unor instrumente de percuţie** [3].

Astfel, instrumentele de percuţie posedă, în general, anvelope dinamice de tip triunghiular, cu atac (*attack*) scurt de maximă intensitate şi cădere (*decay*) abruptă. În Figura 6.2, sunt reproduse câteva modele de anvelope dinamice ale unor instrumente de percuţie. Ele sunt luate din sonograme reale ale acestor instrumente. Se vede că există o diversitate a profilelor datorată diferenţelor de natură dintre instrumente. Se constată că atacurile sunt destul de asemănătoare. Diferenţierea prezentându-se la nivelul "căderii " sunetului, în funcţie de capacităţile diferite ale instrumentelor de a prelungi sunetele.

După modelul acestor anvelope reale, se pot desena amvelope virtuale, utilizabile în procesul de *live electronics* prin aplicarea lor asupra prestaţiei instrumentelor reale. Aceste modele virtuale pot fi făcute să aibă pante mai

diversificate de atacuri, precum şi "căderi" care nu există la instrumentele reale.

6.4 Studiu de caz: *Itinéraires à l'intérieur du son I*

Lucrarea pentru flaut solist, grup concertino (flaut, clarinet, percuţie) şi orchestră, intitulată *Itinéraires à l'intérieur du son I* de Fred Popovici (1990-91) reprezintă o aplicaţie riguroasă a datelor oferite de acustica şi electroacustica modernă, în cazul organizării unui discurs muzical. Cu alte cuvinte, informaţiile furnizate anterior vor fi utilizate aici în mod practic.

6.4.1 Paradigme sonore folosite

Punctul de plecare îl constituie următoarea realitate. Se ştie din psihoacustică faptul că frecvenţa de $16,5Hz$ este pragul percepţiei noastre care diferenţiază sunetele (frecvenţele) de durate (ritmuri): primele se ordonează deasupra acestei frecvenţe, celelalte dedesubt. Avînd în vedere că această frecvenţă este inaccesibilă instrumentelor din orchestră, cu excepţia orgii mari, în compoziţia amintită, va fi folosită octava superioară acesteia, adică $33Hz$. Ea corespunde unui *Do*, ce va fi luat ca reper în tot travaliul următor.

Să considerăm următoarele frecvenţe: $33Hz$, $44Hz$, $49,5Hz$, $55,68Hz$ $74,25Hz$. Ele reprezintă fundamentalele (tonicile) unei *secvenţe armonice canonice*: Do Fa Sol (Do) La Re Sol - [Do], respectiv cadenţa armonică, o paradigmă întâlnită, în mod fix în toată muzica clasică[1]. Ele au fost obţinute în felul următor:

$$\begin{aligned} 33Hz \times 1 &\rightarrow Do \\ 33Hz \times 4/3 &\rightarrow Fa \\ 33Hz \times 3/2 &\rightarrow Sol \\ 74,25Hz \times 3/4 &\rightarrow La \\ 49,5Hz \times 3/2 &\rightarrow Re \end{aligned}$$

Dar, aici, vor fi folosite ca fundamentale ale unor spectre armonice unde vor exista numai armonicele date de numere impare, de tip $2n-1$. Se

[1] Aceste frecvenţe reprezintă componentele paradigmei amintite: prima este fundamentala, cvinta acesteia se obţine prin înmulţirea cu 3/2, iar cvarta prin înmulţirea cu 4/3.

construiesc spectrele de frecvenţe prin *multiplicare* cu mulţimea numerelor impare:

$$33Hz \times \{1, 3, 5, \ldots (2n-1)\} = \{33Hz, 99Hz, 165Hz, \ldots\},$$
$$44Hz \times \{(1, 3, 5, \ldots (2n-1)\},$$
$$49,5HzHz \times \{1, 3, 5, \ldots (2n-1)\},$$
$$55,68Hz \times \{1, 3, 5, \ldots (2n-1)\},$$
$$74,25Hz \times \{1, 3, 5, \ldots (2n-1)\},$$

precum şi spectrele de durate (ritmuri) prin *divizare* cu aceeaşi mulţime a numerelor impare:

$$16,5Hz/\{1, 3, 5, \ldots (2n-1)\} = \{8,25Hz, 4,125Hz, 2,06Hz, \ldots\}$$
$$22Hz/\{1, 3, 5, \ldots (2n-1)\}$$
$$24,75Hz/\{1, 3, 5, \ldots (2n-1)\}$$
$$27,84Hz/\{1, 3, 5, \ldots (2n-1)\}$$
$$37,125Hz/\{1, 3, 5, \ldots (2n-1)\}$$

Nota bene : în practică, valorile obţinute vor fi folosite prin rotunjire. Spre exemplu, 8,25 Hz va deveni 8 Hz, ceea însemnă o optime, la metronom $\text{♩} = 60$.

Duratele - de la "armonicul" 1 (8 secunde) pînă la "armonicul" 127 (aproximativ 1/16 dintr-o secundă) – vor fi atribuite, univoc, armonicelor propriuzise, de la fundamentală, identică cu armonicul 1, $33Hz$, pînă la $16,5Hz \times 127 = 2095Hz$.

Nota bene : şi în această situaţie, practica instrumentală obligă, cum am mai spus, folosirea frecvenţei de $33Hz$; însă, calculul spectral se face pornind de la $16,5Hz$.

Înlănţuirea utilizată în lucrarea *Itinéraires à l'intérieur du son I* este următoarea:

$$Do - Fa - Sol - Do - La - Re - Sol$$

Se presupune că ar urma din nou *Do*, conform paradigmei armonice despre a care am vorbit deja.

În continuare, se identifică componentele spectrale comune ale perechilor formate din fundamentalele succesive din lista anterioară. Pentru aceasta, vom nota spectrul asociat unei fundamentale cu $\{DO\}$, reprezentând mulţimea armonicelor impare ale lui *Do*.

Atunci, intersecţia spectrelor primelor două fundamentale va fi:

$$\{DO\} \cap \{FA\}$$

urmează:

$$\{FA\} \cap \{SOL\}$$
$$\{SOL\} \cap \{DO\}$$
$$\{DO\} \cap \{LA\}$$
$$\{LA\} \cap \{RE\}$$
$$\{RE\} \cap \{SOL\}$$

Să luăm cazul - aproape - cel mai simplu, care este $\{SOL\} \cap \{DO\}$. El este cel mai simplu pentru că două fundamentale aflate în raportul 3/2 (a se vedea modul în care anterior s-a calculat seria fundamentalelor $Do - Fa - Sol - (Do) - La - Re - Sol - [Do]$) oferă cea mai mare mulţime de componente comune, în mod evident, uşor de identificat. Explicaţia se poate găsi în însăşi rezonanţa naturală, unde primul sunet diferit de fundamentală este al 3-lea, care, matematic apare prin multiplicarea cu 3/2.

Exemplul 6.3 *Fie intersecţia* $\{SOL\} \cap \{DO\}$. *Rezultă două mulţimi identice ca şi componente, dar pe poziţii diferite în raport cu fundamentala:*

- $\{3^0_{DO}, 9^0_{DO}, 15^0_{DO}\}$ *: submulţimea armonicelor lui Do pe care le regăsim între armonicele lui Sol*
- $\{2^0_{SOL}, 3^0_{SOL}, 5^0_{SOL}\}$ *: submulţimea armonicelor lui Sol pe care le regăsim între armonicele lui Do*

unde: n^0_{NOTA} *reprezintă armonicul* n *al notei* $NOTA$.

◇

Exemplul 6.4 *Fie intersecţia* $\{DO\} \cap \{FA\}$. *Rezultă două mulţimi identice ca şi componente, dar pe poziţii diferite în raport cu fundamentala:*

- $\{4^0_{DO}, 12^0_{DO}, 20^0_{DO}\}$ *: submulţimea armonicelor lui Do pe care le regăsim între armonicele lui Fa*
- $\{3^0_{FA}, 9^0_{FA}, 15^0_{FA}\}$ *: submulţimea armonicelor lui Fa pe care le regăsim între armonicele lui Do*

◇

Trebuie menţionat faptul că, pe măsură ce urcăm pe scara spectrelor, componentele crescând numeric, cele comune între două spectre cresc şi ele numeric, mulţimea de intersecţie mărindu-se constant. În zona cea mai înaltă, numită *zona reziduală*, practic se asistă la o convergenţă a spectrelor datorată faptului că o diferenţă mică între două armonice de frecvenţă foarte mare devine irelevantă.

Toate aceste date au consecinţe în diversele ipostaze de transformare în timp real (*live electronics*) aplicată discursului sonor. Să presupunem, de exemplu, că anvizajăm chiar reliefarea trecerii de la un spectru la altul prin două "culori" diferite aplicate mulţimii comune, ce are o anumită pondere faţă de fundamentala X şi alta faţă de fundamentala imediat succesivă, Y. Atunci, pe o anumită porţiune a spaţiului sonor, se va aplica un anume tratament de transformare în timp real a sunetului, iar pe porţiunea următoare unul diferit.

6.4.2 Sintagmatizare 1

Menţionăm, încă odată, faptul că lucrarea *Itinéraires à l'intérieur du son I*, a cărei primă pagină este reprodusă în Figura 6.4, este concepută pentru un instrument obligat/solist, flautul, la care se adaugă un grup restrîns de instrumente ce înconjoară solistul (numit, în muzica clasică, concertino), ce constă din trei instrumente: percuţie, un al doilea flaut şi clarinetul. În jurul tuturor se află un grup orchestral cameral. Atribuţiunile fiecărei entităţi indicate sunt diferite. Astfel, numai flautul solist şi instrumentele din grupul concertino vor intra sub incidenţa tratamentului semnalului sonor în timp real (*live electronics*).

Să racapitulăm cele ce au fost expuse anterior şi să clarificăm, definitiv, un aspect de strategie decizională componistică.

Este vorba despre prima secţiune a primei părţi a lucrării. Aceasta cuprinde spectrele frecvenţelor indicate la 6.4.1 (spectrele, s-a menţionat, alcătuite din multipli cu numere impare ale acestor fundamentale). Succesiunea respectivă este marcată – la schimbarea fundamentlelor - de apariţia subliniată a componentelor comune. Cu alte cuvinte, ultima porţiune (componentele impare ale fundamentalei F_1) şi prima porţiune a noii fundamentale (componentele impare ale lui F_2). Schematic, lucrurile se prezintă ca în Figura 6.3. Cele două coloane juxtapuse reprezintă, cum am arătat, spectrele

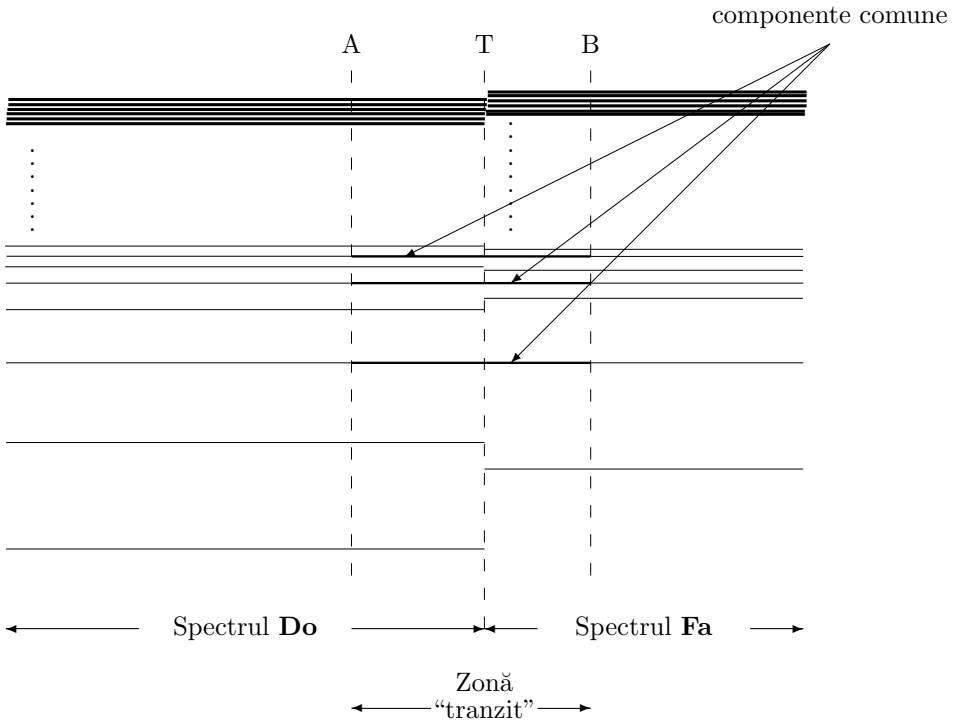


Figura 6.3: **Reprezentarea schematică a tranziţiei între spectre (respectiv Do şi Fa).**

de armonice impare ale fundamentalelor Do (33Hz) şi Fa (44Hz). În acelaşi timp este marcată, în zona de "tranzit", prin linii îngroşate seria componentelor comune ale acestor spectre. Este reliefat faptul că trecerea de la un spectru la altul se face prin aportul sonor al instrumentelor din *concertino* (percuţia mai ales) care intonează aceste sunete de intersecţie.

Pentru edificarea celor care vor să aprofundeze muzical problema înfăţişată, reproducem în Figura 6.4 prima pagină a partiturii. Se constată distribuţia la diverse instrumente a spectrului de impare ale lui Do şi apariţia, la un moment dat, la percuţie – un *glockenspiel* – a sunetelor de intersectie între speectrele lui Do şi ale lui Fa.

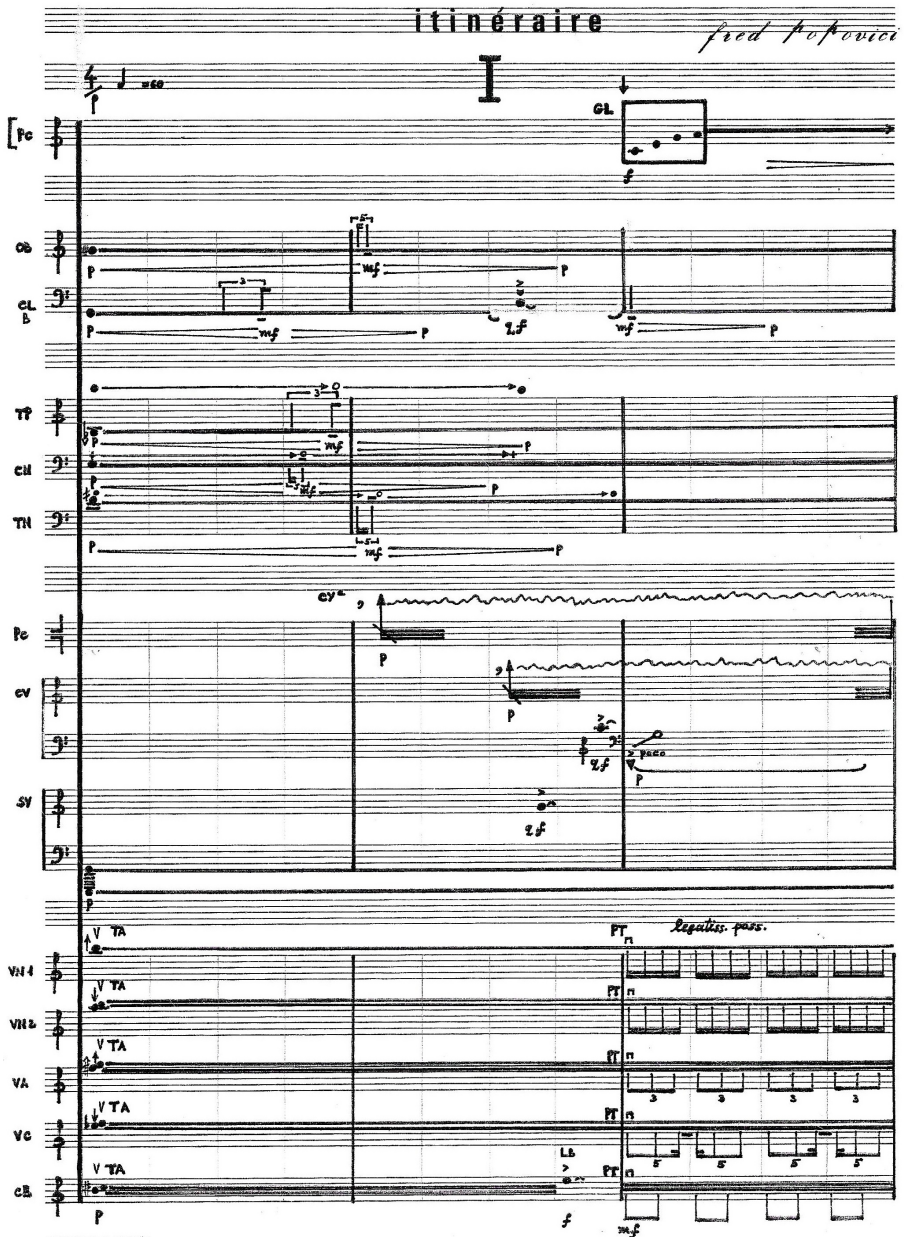


Figura 6.4: **Prima pagină din *Itinéraires à l'intérieur du son I.***

6.4.3 Sintagmatizare 2

Secţiunea principală – de fapt, centrul lucrării – a părţii I-a este ocupată de o structură destul de complexă, ce simulează, în linii generale, procedeul binecunoscut, din eleectroacustică, al *buclei de re-injecţie*. În substanţă, este vorba, în cazul acestei bucle, de reiterarea multiplă a unei structuri sonore ce se degradează şi/ sau se disipează pe măsură ce această reiterare progresează.

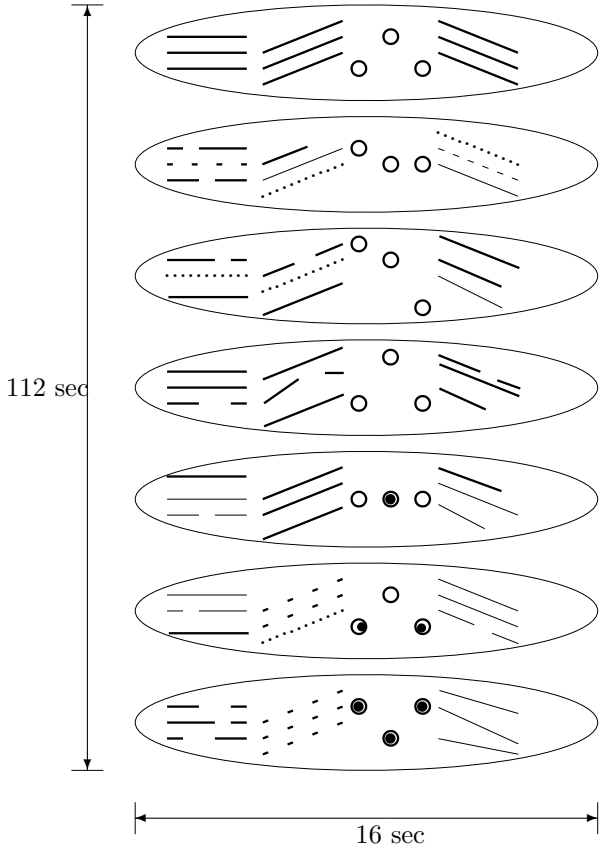


Figura 6.5: **Simularea buclei de reinjecţie. (Secţiunea centrală a părţii I).**

De exemplu, în cazul unei înregistrări analogice, mecanice, pe bandă de magnetofon înregistrarea şi copierea de pe această bandă pe o altă bandă produce, după câteva reluări, distorsiuni evidente, imediat perceptibile, ale

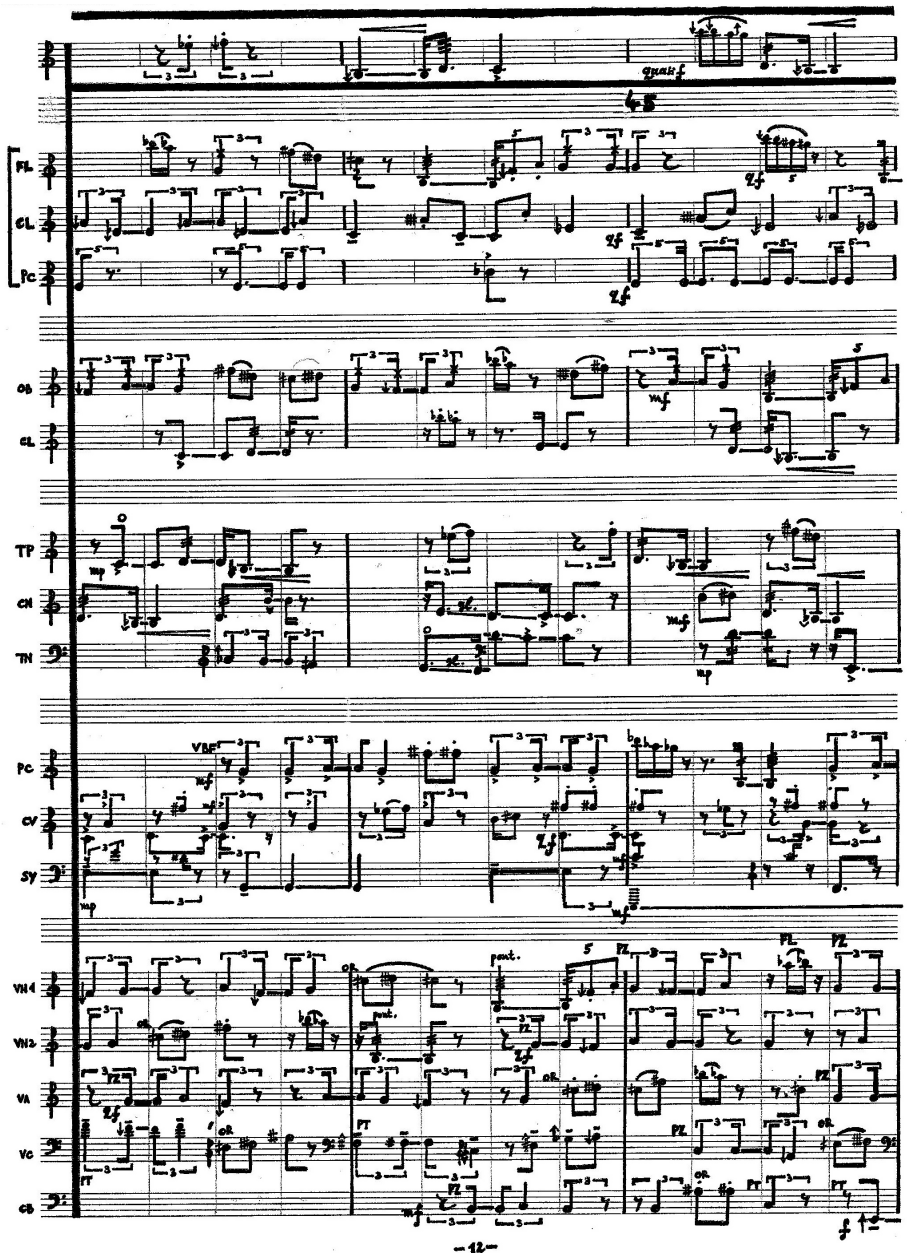


Figura 6.6: **A 12-a pagină din *Itinéraires à l'intérieur du son I.***

materialului înregistrat. Trebuie arătat faptul că acest procedeu pur tehnic ilustrează şi subliniază foarte bine logica intrinsecă a oricărui discurs sonor care – derulându-se pe axa negentropică a timpului – antrenează, în desfăşurarea lui, o ireversibilă transformare a parametrilor conţinuţi.

În *Itinéraires à l'intérieur du son I*, organismul sonor (sau structura, dacă am folosi un limbaj mai convenţional) ce este supus procedeului descris mai sus este realizat în felul următor:

- are o durată de (aproximativ) 16 secunde: în termeni muzicali aceasta înseamnă 4 măsuri de câte 4 timpi fiecare, la MM♩= 60
- este alcătuită din suprapunerea a trei contururi monodice (melodice), primul extras din câteva armonice ale lui Do (33 Hz), al doilea, similar, din armonicele lui Fa (44 Hz) şi al treilea din cele ale lui Sol (49,5 Hz). Cele trei contururi sunt repartizate, în ordine, flautului solist şi grupului concertino (flaut, percuţie şi clarinet).

Pe parcursul a 7 "episoade" (incluzând şi pe cel iniţial, descris anterior), având o durată de 16 secunde multiplicate cu 7 (aproximativ două de minute), organismul sonor iniţial se transformă, se distorsionează şi disipează continuu: toţi parametrii sonori sunt "fractalizaţi", în accepţiunea generică a termenului.

În Figura 6.5, este prezentat vizual procesul ce a fost enunţat anterior. Fiecare "episod" este arătat ca o elipsă simbolizând anvelopa virtuală a acestuia. În interiorul elipsei, cele trei contururi melodice sunt desenate prin module grafice alese arbitrar numai pentru puterea lor de sugestie. Transformările continue, deja menţionate, sunt reprezentate prin modificări continue ale celor trei conducte melodice. (Fiecare conduct melodic, în interiorul unui "episod", este alcătuit dintr-o linie orizontală, una cu o înclinare spre dreapta, un cerculeţ şi o linie cu înclinare spre stânga.) Faptul că toţi parametrii ce alcătuiesc aceste linii melodice sunt alteraţi în mod neregulat şi continuu este sugerat de felul în care, de la un episod la altul, fiecare linie şi cerculeţ variază ca formă tot continuu.

Capitolul 7

Aplicarea tehnicilor de live electronics

În literatura de specialitate, prin tehnicile de *live electronics* se înţelege totalitatea procedeelor prin care se aplică diverse transformări sunetului în chiar timpul producerii lui. În felul acesta, se înţelege că *live electronics* se separă total de tehnicile electronice pre-preparate; prin ele înţelegem înregistrări făcute înainte de performanţă, pe bandă de magnetofon şi, mai recent, pe alte suporturi electronice, care sunt derulate în timpul execuţiei lucrării. Din acest motiv, tehnicile *live electronics* sa înrudesc – şi chiar se confundă, până la un anumit punct – cu impovizaţia realizată informaţional; aceasta înseamnă că produsul *live* este transformat în timp real printr-un instrument oarecare chiar de către interpret (un astfel de instrument este pedala de care dispune instrumentistul, pe care o apasă în timp ce cântă la instrumentul său, semnalul fiind transmis unui calculator ce produce sunete în funcţie de un program prestabilit).

7.1 Tehnici de live electronics

În linii generale, tehnicile de *live electonics* însemnă apelul la:

procedee paradigmatice; sunt cele care, de obicei, pot fi aplicate indeterminat la orice tip de realizare cu *live electronics*; ele sunt:

- întârzierea semnalului (*delay*)
- reverberaţia (*reverb*)

- ecoul (*echo*)
- panoramarea
- efectul Doppler

procedee sintagmatice; sunt toate procedeeele care fac apel la mijloace elaborate, bazate pe programe, precum:

- filtrele
- sinteza în timp real a sunetului
  - adiţională
  - granulară
- modulaţiile
  - *ring modulation*
  - modulaţia de frecvenţă
  - modulaţia de amplitudine

7.1.1 Procedee paradigmatice

Delay Procedeul delay-ului, pe care, uneori, în literatura de specialitate, îl întâlnim sub titulatura de efect (alături de celelalte paradigmatice), se recomandă de la sine: înseamnă, pur şi simplu, că semnalul emis de o sursă oarecare este decalat, în timp, faţă de el însuşi. Eficienţa lui este funcţie de intervalul de timp ales pentru a-l realiza.

Reverberaţia este, într-adevăr, un efect ce s-a folosit, în înregistrări, încă din timpuri imemoriale. El ajută sunetul să străbată mai plin în audienţă. Există o mare varietate de metode (simple) pentru a-l realiza.

Ecoul, la rîndul său, este foarte solicitat, atît în înregistrări, cît şi în execuţiile muzicale în timp real.

Panoramarea este un efect (procedeu) ce se referă la acţiunea sunetului la nivelul difuzoarelor. Avînd mai multe aparate de acest fel, sunetul poate fi distribuit în foarte multe modalităţi în fiecare difuzor în parte, ca şi la nivelul raporturilor dintre acestea, în legătură directă cu scopul acustic urmărit de acţiunea asupra sunetului.

Efectul Doppler este un procedeu mult mai complicat decît celelalte, mult mai greu de realizat şi la care, majoritatea rezultatelor obţinute este mai degrabă la nivel metaforic, o simulare aproximativă a ceea ce se întîmplă în realitate. Se ştie că o sursă sonoră produce senzaţia de creştere a înălţimii (frecvenţei) sale cînd se apropie şi efectul contrar (scăderea înălţimii) cînd se îndepărtează. Instrumentele unui ansamblu ce execută o lucrare muzicală se află, tot timpul, în poziţii fixe şi de aceea, spuneam, efectul Doppler poate fi doar vag aproximat, exclusiv prin prin procedee electroacustice.

7.1.2 Procedee sintagmatice

Filtrele sunt procedee care acţionează asupra componentelor spectrale ale unui sunet, realizînd distorsiuni semnificative ale acestuia. Există, în primul rînd, filtre naturale (să le spunem aşa), care sunt toate tipurile de surdine, atît pentru instrumentele de suflat din alamă – trompeta, cornul, trombonul, tuba – cît şi pentru instrumentele de corzi – vioara, viola etc. Cu puţină fantezie, se pot confecţiona şi surdine pentru suflătorii de lemn – flautul, oboiul etc. Filtrele, pe de altă parte, sunt procedee abstractizabile, teoretic concepute, de acţiune asupra semnalului sonor. În electroacustică, se întîlnesc trei tipuri de filtre: *low-pass filter* (filtru trece-jos), *high-pass filter* (filtru trece-sus) şi *band-stop filter* (filtru opreşte-bandă). Acţiunile lor sunt: primul taie din frecvenţele înalte ale spectrului, al doilea, invers, din cele joase, iar al treilea taie pe o anume fîşie spectrul respectiv.

Sinteza adiţională, adică sinteza prin adunare (ori, invers, prin scădere) a fost prezentată în secţiunea 6.2. şi nu mai revenim aspra ei.

Sinteza granulară este un procedeu mult mai recent şi se realizează cu ajutorul calculatorului. Compozitorul grec Yannis Xenakis a avut, primul, intuiţia realizării unei structuri sonore foarte dense, alcătuită, însă, din nenumărate particule infinitezimale (**granule** sonore cu durate variind între 1 ms şi 50 ms), în compoziţiile sale din anii 1950, *Metasthasis* şi *Pithoprakta*. Mai tîrziu, către sfârşitul secolului trecut, mai mulţi teoreticieni şi compozitori au preluat ideea lui Xenakis şi au dezvoltat-o conceptual. În esenţă, sinteza granulară are puternice similitudini cu metoda eşantioanelor (a *samplerurilor*), bazîndu-se pe crearea unor fragmente extrem de mici şi extrem de mobile ale parametrilor sonori.

Ring-modulation este o modalitate de lucru prin care se produc efecte reciproce între două semnale sonore, astfel încît suma şi diferenţa dintre valorile lor frecvenţiale să realizeze straturi inarmonice. Este înrudită, într-un fel, cu modulaţia de amplitudine.

Celelalte două tipuri de modulaţie sunt procedee electronice consacrate.

7.2 Studiu de caz (continuare)

Vom relua în această secţiune problemele abordate, din punctul de vedere al strategiei compoziţionale, pentru a le transla în domeniul interpretării extinse, pe care-l reprezintă procedeele de *live electronics*. Este vorba de lucrarea lui Fred Popovici, *Itineraires à l'interieur du du son*.

7.2.1 Procesarea live electronics a tranziţiei între spectre

Cum s-a mai spus, tot ceea ce a fost arătat în paragraful 6.4.2, procesualitatea componistică expusă, trebuie să fie ranforsată şi mai limpede prin introducerea procedeelor de tratament în timp real al sunetului, recte, de către instalaţia de *live electronics*. Porţiunea ce reprezintă zona de "tranziţie" este cea care va fi supusă acestui tratament. Cu alte cuvinte, doar instrumentele din grupul concertino (percuţia, flautul şi clarinetul) vor fi transformate în timp real.

Întrucât tranziţia de la spectrul lui Do la cel al lui Fa, la începutul lucrării, este realizată doar de percuţionistul din grupul concertino, pe instrumentul numit *glockenspiel*, ne vom concentra asupra procedurii de *live electronics* aplicată asupra acestuia. *Glockenspiel*-ul intervine după un anumit interval de timp (de la punctul notat cu A în Figura 6.3) după ce orchestra a produs spectrul lui Do şi continuă să intoneze componentele comune, după schimbarea fundamentalei în Fa (la momentul T) până la momentul B, când încetează. În Figura 7.1 am reprezentat numai componentele comune celor două spectre ce se succed şi pe care le însumează *glockenspiel*-ul.

Se lucrează la două niveluri:

- cel al unor procedee din zona numită paradigmatică
- cel al transformărilor mai complicate, sintagmatice.

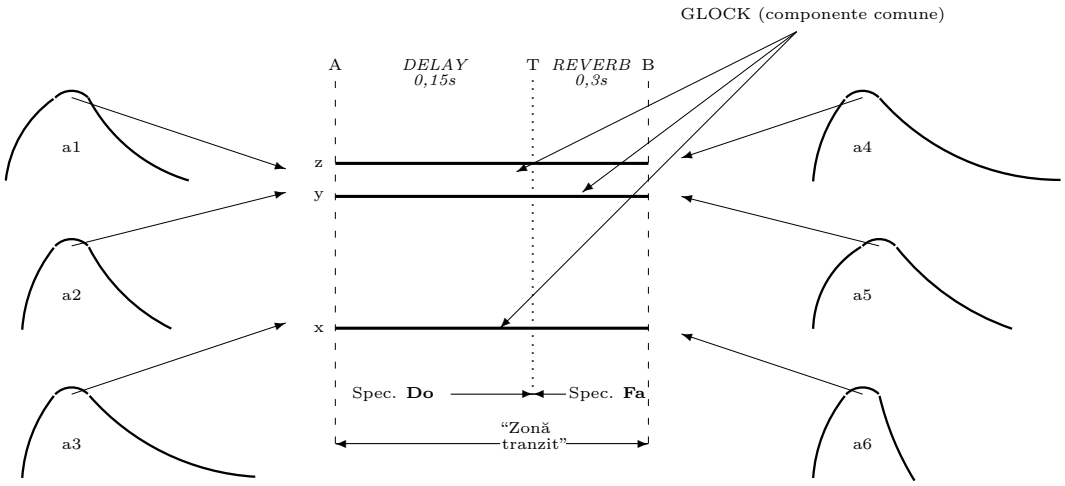


Figura 7.1: ***Live electronics* pentru tranziţia dintre spectrele lui Do şi Fa.**

Primuul nivel se aplică în felul următor: porţiunii A-T, din prestaţia instrumentului, i se aplică un *delay* de $0,15sec$, iar în porţiunea T-B *delay*-ul este înlocuit printr-un *reverb* de $0,3sec$.

Al doilea nivel produce modificarea anvelopelor de atac al sunetului. Să presupunem că celor trei componente comune – x, y, z – li se aplică în portiunea A-T anvelopele **a1**, **a2** şi **a3**, iar în porţiunea T-B cele trei anvelope sunt înlocuite cu **a4**, **a5** şi **a6** (vezi Figura 7.1). Toate cele 6 anvelope pot fi "variaţiuni" ale unor anvelope luate din înregistrări ale *glockenspiel*-ului. Pe de altă parte, există şi posibilitatea ca ele să fie desenate după anvelopele reale descoperite în înregistrări ale altor instrumente de percuţie.

7.2.2 Procesarea live electronics a buclei de reinjecţie

Totalitatea procedeelor care au generat forma muzicală vor fi utilizate în mod concret pentru extinderea execuţiei sonore prin *live electronics*. Aceasta înseamnă că vor exista două paliere de lucru, cel prin care se aplică şi în tehnologia *live electronics* un filtru/amplificator şi un al doilea palier prin care se acţionează similar în ceea ce priveşte un generator de zgomot prin

"tasare" (vezi sistemul FAZT descris în 5.3.3).

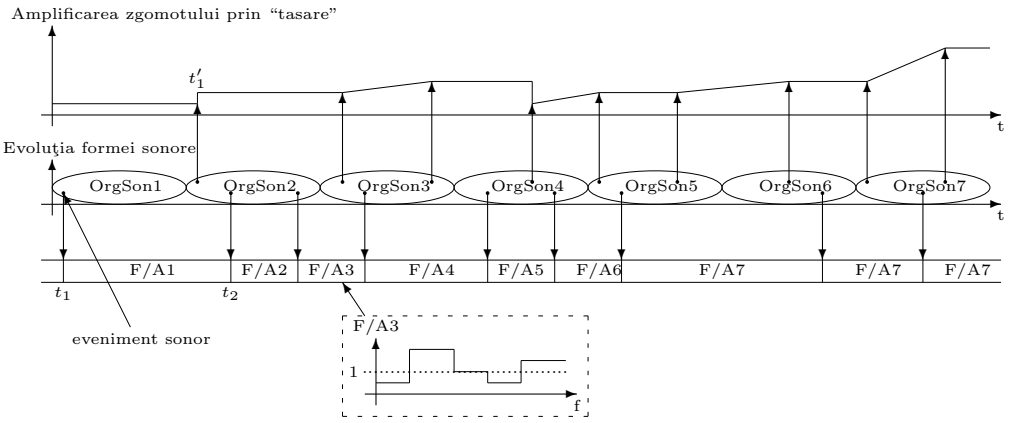


Figura 7.2: **Interacţia dintre execuţia formei sonore şi FAZT.**

În Figura 7.2 sugerăm modul în care procedele de *live electronics*, bazate pe dispozitivul FAZT, se aplică practic formei muzicale realizare în secţiunea 6.4.3. Evoluţia formei sonore este reprezentată prin aceleaşi elipse – `OrgSon1, ... OrgSon7` – care sunt ordonate liniar. Deasupra şi dedesubtul lor au fost reprezentate cele două procesualităţi denumite *amplificarea zgomotului prin "tasare"*, respectiv înlănţuirea profilelor diferite ale filtrului/amplificator – `F/A1 ... F/A9`. Punctele din `OrgSon`-uri din care emerg, în sus şi in jos, săgeţile sunt micoevenimente indicate în partitură. Microevenimentele declanşează procesul de modificare fie a amplificării generatorului de zgomor prin "tasare", fie a profilului filtrului/amplificator. Pentru edificare, am exemplificat, în figură, profilul F/A3.

Pentru realizarea efectivă a unei execuţii în concert, toate profilele aferente celor două niveluri sunt configurate în prealabil, astfel încât ele pot fi apelate prin simple comenzi de tip `next`.

Compoziţia spectrală a zgomotului de "tasare" poate fi specificată în mai multe moduri. Spre exemplu, poate fi indicată ca frecvenţă de "tasare" o anumită frecvenţă, constantă de a lungul întregului proces. Sau, o altă posibilitate, este selectarea în timp real a fundamentalei cu frecvenţa minimă.

Capitolul 8

Arhitectură *live electronics*

În cele ce urmează, vom propune o modalitate de creştere a eficienţei unui sistem *live electronics*. Nivelul actual al cercetărilor şi realizărilor din domeniul *live electronics* poate fi sintetizat în felul următor: execuţia unei lucrări muzicale este urmărită de o echipă de asistenţă tehnică, care operează asupra dispozitivelor de transformare/generare a semnalului. Concret, echipa urmăreşte desfăşurarea microevenimentelor execuţiei şi intervine, activând funcţii presetate, ce pot fi incluse în ceea ce s-a spus anterior referitor la procedee paradigmatice sau sintagmatice.

Tehnologiile IT curente ne permit să elaborăm sisteme mult mai eficiente, ceea ce se traduce prin mărirea complexităţii şi fineţei travaliului, ca şi prin creşterea vitezei de intervenţie a procedeelor *live electronics*. Un alt efect, pe plan economic, este reducerea echipei de asistenţă tehnică, prin asumarea de către un sistem automat a acţiunilor acesteia. Implicit, prin această strategie, se doreşte contactul direct al compozitorului cu "instrumentul" care-i traduce intenţiile. Astfel, compozitorul nu mai are nevoie de o calificare avansată în domeniul IT. Un efect suplimentar, de neignorat, este eliminarea intervalelor mari de timp de lucru într-un centru electroacustic.

În acest moment putem vorbi despre propunerea proprie a unei arhitecturi de *live electronics*. Pentru a ilustra modul de funcţionare a acestei arhitecturi, pornim de la aplicarea practică a *Sintagmatizării 2*, parte a lucrării *Itinéraires* (vezi 7.2.2). În Figura 7.2 forma sonoră reală, bazată pe partitură, este acompaniată de execuţia unor procedee de *live electronics* declanşate de micoevenimente existente şi identificate în această formă. Evenimentele sunt marcate prin puncte din care emerg sageţi către profilele caracterizând procesările *live electronics* folosite (în acest exemplu): zgomorul de "tasare" şi

filtrul/amplificator.

Problema esenţială în funcţionarea arhitecturii propuse este stabilirea concordanţelor, mai mult sau mai puţin aproximative, dintre partitură şi execuţia ei. Aceste concordanţe sunt necesare pentru identificarea punctelor remarcabile ce declanşează diferitele etape ale procesului *live electronics* (vezi Figura 7.2).

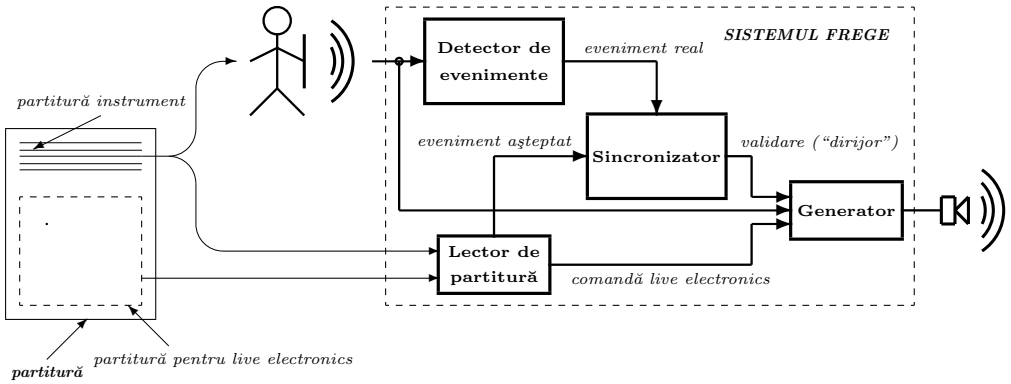


Figura 8.1: Sistemul FREGE de *live electronics*

Arhitectura propusă este reprezentată schematic în Figura 8.1. În linii mari, avem trei componente principale: o partitură, un interpret care o execută şi sistemul de *live electronics* propus (pe care-l denumim FREGE).

Partitura este constituită din ceea ce putem numi *pre-partitură* (forma standard a unei partituri) şi din "partitura" după care se execută procesarea *live electronics*. Partitura *live electronics* conţine indicaţiile pentru acest demers, sub forma unor simboluri ce definesc un limbaj specific.

Sistemul *live electronics* se constituie din următoarele componente:

Lector de partitură: cel care urmăreşte simultan simbolurile din cele două sub-partituri

Detector de evenimente: cel care recepţionează convertirea în semnal sonor a semnelor din partitură şi regenerază partitura sub formă simbolică

Sincronizator: dectectază producerea microevenimentelor sonore stabilite ca fiind remarcabile în sub-partitura *live electronics*

Generator: are trei intrări:

- de la **Sincronizator**, de la care primeşte un semnal de declanşare a unui transformări *live electronics*; deci primeşte comanda de ***când***
- de la instrumentist, de la care primeşte semnalul ce va fi transformat; deci primeste ***ce*** se transformă
- dela **Lector de partitură**, de la care primeşte transformarea sub forma unei secvenţe simbolice de configurare; deci primeşte comanda referitor la ***cum*** se transformă.

Referindu-ne la procesul reprezentat în Figura 7.2, avem următoarea desfăşurare:

- iniţial atât **Generator de zgomot prin "tasare"** cât şi **Filtru/amplificator**-ul (vezi Figura 5.6) sunt preprogramate cu valori de start
- la momentul t_1 este detectat, de către **Sincronizator**, un microeveniment care determină modificarea parametrilor **Filtru/amplificator**-ului prin transferul bloclui de date F/A1 **Generator**-ului de către blocul **Lector de partitură**
- la momentul t_1' este detectat, de către **Sincronizator**, un alt microeveniment care determină modificarea parametrilor generatorului de zgomot
- la momentul t_2 este detectat, de către **Sincronizator**, un microeveniment care determină modificarea parametrilor **Filtru/amplificator**-ului prin transferul bloclui de date F/A2 **Generator**-ului de către blocul **Lector de partitură**
- ...

Realizarea acestei arhitecturi presupune un proiect cu următoarele etape: definirea limbajului "partiturii" *live electronics*, definirea şi implementarea componentelor sistemului FREGE.

Definirea limbajului "partiturii" *live electronics* se face pe două niveluri. Un prim nivel se referă la simbolurile şi parametrii pe care-i scriem direct în partitură. Spre exemplu: `delay(200), reverb(300), F/A5ZT(300, 2), F/A5`. Formulele anterior listate sunt intrepretate diferit. Primele trei conţin informaţia completă asupra procesului la care se referă. Într-adevăr, `delay(200)` se traduce prin întârziere de 200 ms. Ultima formulă se referă la un proces caracterizat complex printr-o listă structurată de parmetri. Spre exemplu:

```
(F/A5 ((0 300 0,8)(300 500 2,3) ...  (1200 1350 0,3))
```

este interpretat ca descriind o anvelopă de frecvenţă cu următoarele caracteristici: de la 0Hz la 300 Hz amplificarea este 0,8, de la 300Hz la 500 Hz amplificarea este 2,3, ş.a.m.d.

Existenţa unei astfel de liste, în anumite cazuri, impune cel de al doilea nivel, inaparent, al "partiturii" *live electronics*. Unde apare, totuşi, acest nivel? În programele care descriu funcţionarea blocului **Generator**.

Între modulele sistemului FREGE există o diferenţiere de complexitate în structurarea lor. Cel mai dificil de realizat este blocul **Detector de evenimente** deoarece trebuie să pornească de la sunetele emise de interpret şi să reconstituie pre-partitura. Este de fapt operaţia inversă execuţiei. Această operaţie este întotdeauna aproximativă, întrucât nu există interpretare perfect suprapusă datelor pre-partiturii. Sarcina blocului **Sincronizare** este să estimeze gradul de concordanţă între semnal şi partitură, pentru a determina momentul efectiv de declanşare a unei transformări. Blocul **Generator** este implementat de unul dintre programele standardizate de *live electronics* (spre exemplu: Pure Data, MAX/MSP), care-şi primeşte input-ul de la **Sincronizator**, **Lector de partitură** şi semnalul emis de interpret.

Avantajele pe care le estimăm prin realizarea arhitecturii FREGE ar fi: creşterea eficienţei în aplicarea procedeelor *live electronics* şi *ipso facto* ajutorul acordat compozitorului în travaliul său. Un alt avantaj este simplificarea relaţiei dintre creator şi ansamblul de interpreţi.

Partea III

ANEXE

Anexa A

ANTLRWorks

Autor: ***Robert Alexandru Dobre***

ANTLRWorks (ANTLR ANother Tool for Language Recognition) este un mediu pentru dezvoltarea de gramatici scris de Jean Bovet. Acesta este format dintr-un editor cu ajutorul căruia se pot defini gramaticile şi un asistent de interpretare ("interpreter" în limba engleză) care, pe baza unei expresii introduse, va genera arborele de parsare şi va indica dacă expresia face sau nu parte din gramatica descrisă în editor. Avantajul major al acestei aplicaţii este că oferă o interfaţă grafică suficient de sugestivă, utilizatorul fiind nevoit să înveţe numai modul în care se definesc gramaticile. Introducerea regulilor, terminalelor şi neterminalelor facându-se într-un limbaj de nivel înalt, lucrul cu aplicaţia este mult simplificat. În continuare, vom prezenta, cu ajutorul câtorva exemple, cum se poate folosi ANTLRWorks, pentru a defini gramatici şi a parsa expresii.

A.1 Instalare

Înaintea instalării ANTLRWorks, este necesar să se instaleze Java (`http://java.com/en/download/index.jsp`). În continuare, se va descărca programul de la următoarea adresă web: `http://www.antlr3.org/works/`.

A.2 Editorul

Aplicaţia imediat după lansare va arăta ca în imaginea din Figura A.1.

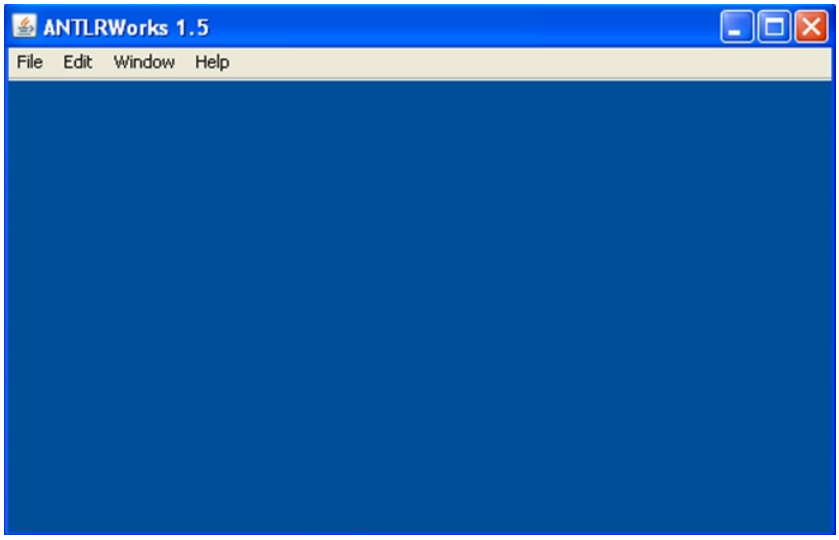


Figura A.1: **ANTLRWorks prima lansare.**

Pentru a introduce o nouă gramatică, se va selecta `New...` din meniul `File`. Din noul meniu se va selecta `ANTLR 3 Grammar (*.g)`. În fereastra `New Grammar Wizard` se va introduce un nume pentru gramatică în câmpul `Grammar Name` şi se va selecta `OK`. Dacă, de exemplu, a fost introdus numele `test`, fereastra ar trebui să arate ca cea din Figura A.2.

Se observă că aplicaţia a scris automat prima linie de cod şi anume: `grammar test;`. Se va salva fişierul cu acelaşi nume ca cel al gramaticii. În continuare, putem începe să definim reguli, terminale şi neterminale.

Neterminalele se introduc cu litere mici, terminalele se scriu între ghilimele simple (exemplu: terminalul a va fi introdus ca `'a'`). Pentru a evita eventualele confuzii, este recomandat să se introducă numele unui neterminal urmat de litera `n` (**n**eterminal) sau marcat într-un alt fel ales de utilizator. Neterminalele pot fi scrise şi ca grupuri de litere şi/sau cifre (de exemplu: `a` sau `an` sau `neterminalula`). Acest lucru va face mai uşoară citirea arborelui de parsare (programul nu afişează diferit terminalele şi neterminalele în arbore, utilizatorul trebuie să poată distinge între ele. Dacă avem, de exemplu, neterminalul a care poate trece în terminalul `'a'`, arborele poate fi mai greu de citit, mai ales în procesul de debugging).

Regulile au următoarea sintaxă:

```
neterminal :  expresie_1 | expresie_2 | ...  | expresie_n ;
```

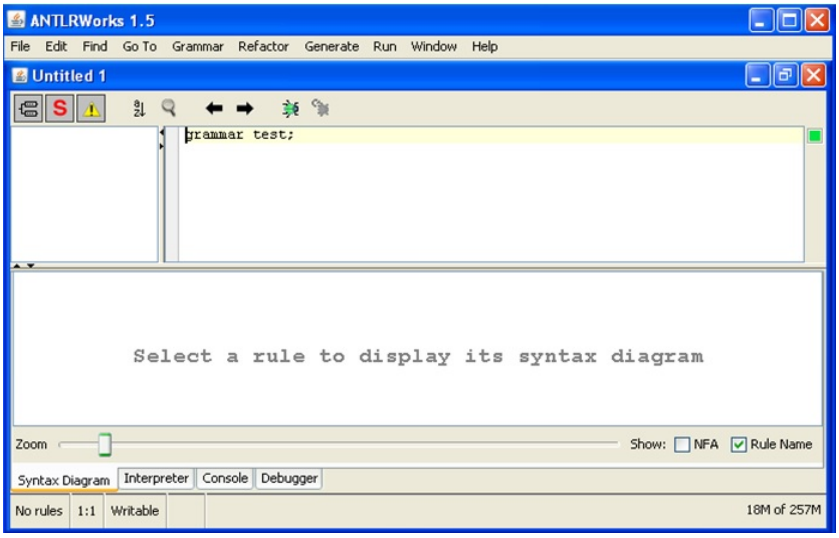


Figura A.2: **Editorul pentru gramatica cu numele "test".**

unde: `expresie_i, i=1...n`, poate fi un neterminal, un terminal sau o combinaţie de terminale şi neterminale. Combinaţiile se scriu introducând minimum un spaţiu după fiecare terminal/neterminal (vezi secţiunea *Reguli recursive* pentru un exemplu). Se observă că terminalele şi neterminalele sunt definite în cadrul regulilor.

Pentru exemplificarea celor descrise mai sus, vom scrie o gramatică ce poate genera o singură literă, `'a'` sau `'b'`. Regulile, conform notaţiilor din curs, sunt următoarele: `S` $\rightarrow$ `A,a` şi `A` $\rightarrow$ `b`. Se poate observa că putem scrie totul sub forma `S` $\rightarrow$ `a,b` fără a avea nevoie de neterminalul `A`, dar exemplul este pur didactic.

Aşadar vom introduce regulile după cum urmează:

```
sn :  an | 'a'; // S → A,a
an :  'b'; // A → b
```

Fereastra ar trebui să arate ca în Figura A.3.

În partea de jos a ferestrei programului, regulile se pot observa sub formă de diagramă. Pentru a observa o regulă în această formă, trebuie să punem cursorul undeva în codul care descrie regula (în cazul de faţă cursorul este înainte de neterminalul 'a', în prima regulă, deci obţinem reprezentarea acestei reguli sub formă de diagramă). Se mai observă că neterminalele şi terminalele sunt scrise cu culori diferite.

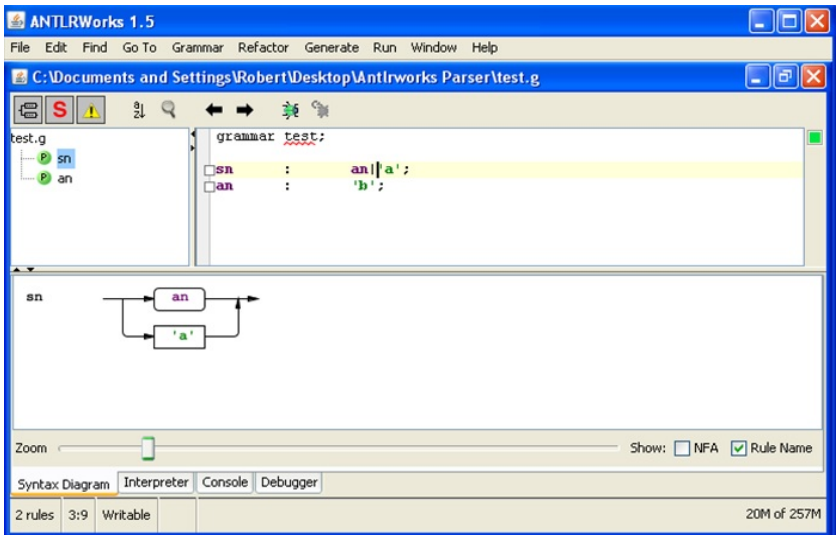


Figura A.3: **Gramatica "test" cu cele două reguli introduse.**

A.3 Asistentul pentru interpretare

Pentru a găsi dacă anumite expresii pot fi generate cu ajutorul gramaticii introduse, vom folosi asistentul pentru interpretare. Pentru aceasta, trebuie să selectăm tab-ul Interpreter din partea de jos a ferestrei programului.

Zona în care erau afişate diagramele s-a împărţit în două. În partea din stânga se va introduce expresia de evaluat, iar în partea din dreapta va fi afişat arborele de parsare. Pentru a parsa o expresie, după introducere se va selecta neterminalul rădăcină al arborelui de parsare (din drop-down list-ul indicat de caseta 1) şi se va apăsa butonul de start (indicat de caseta 2). Va fi afişat arborele de parsare, neterminalul `sn` a trecut direct în terminalul `a`. Puteţi încerca să parsaţi expresia ce conţine numai litera `b`. Ar trebui să se obţină arborele de parsare pornind din neterminalul sn care va trece în neterminalul an care, la rândul său, va trece în terminalul `b`.

Pentru o parsare corectă, în vârfurile arborelui de parsare se va obţine chiar expresia introdusă. Dacă expresia este incompletă, parsarea nu a putut fi efectuată.

Acestea sunt mecanismele de bază ale ANTLRWorks. În continuare, vom prezenta câteva instrumente ce vor fi utile în scrierea gramaticilor mai complexe.

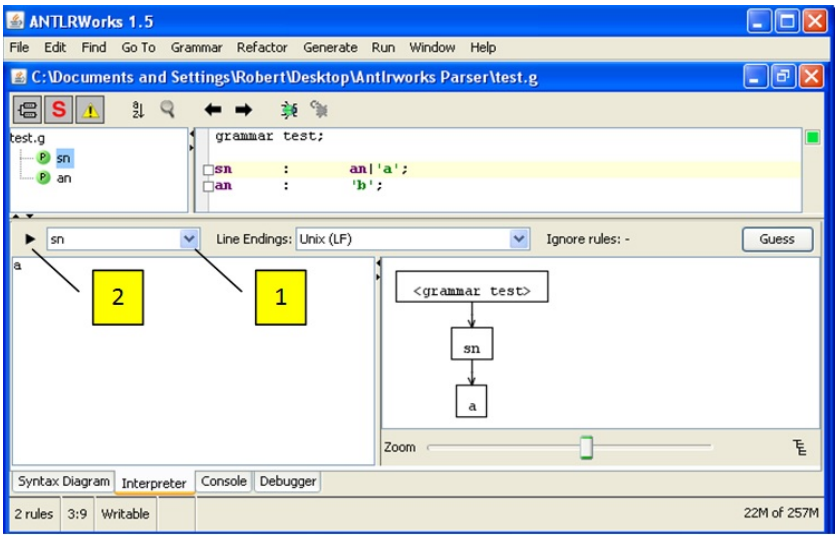


Figura A.4: **de interpretare şi arborele de parsare pentru expresia 'a'**

A.4 Reguli recursive

ANTLRWorks nu suportă în scriere directă reguli recursive. De exemplu, `A` → `Ab` introdusă ca

```
an :  an 'b';
```
// de observat cum în combinaţia neterminal-terminal se foloseşte un spaţiu

ar fi semnalizată ca având o eroare, şi anume că este "left-recursive". Programul poate rescrie regulile automat pentru a nu mai fi recursive. În acest sens, după ce au fost introduse toate regulile, dacă există reguli recursive, se va selecta opţiunea `Remove All Left Recursion` din meniul `Refactor`.

Un alt exemplu de regulă recursivă, pentru familiarizarea cu introducerea combinaţiilor de terminale/neterminale:
`an : an bn 'a' | abc123 an;` //se vor observa spaţiile şi faptul că şi neterminalele pot fi reprezentate de grupuri de simboluri.

A.5 Lookahead depth

Un instrument important în ANTLRWorks este "lookahead depth"-ul. Mai multe informaţii despre acesta se pot găsi la adresa: `http://www.antlr2.org/doc/options.html#k` .

A.6 Exemple utile

Folosind şi aceste două instrumente avansate, se poate defini gramatica dedusă pe baza operelor lui Mozart, prezentată în curs. Codul final, după eliminarea automată a recursivităţii, este urmatorul:

```
grammar mozart;

options{
k=1; //lookahead depth
}

//setul de reguli
s : a b a;
a : ('42') (a c a)*;
b : (c b | '31') (c)*;
c : ('1111') (d)*;
d : '2222';
```

Alt exemplu prezentat în curs este descris în ANTLRWorks cu urmatorul cod:

```
grammar pag34;
//generare expresii tip ...a+a+a+a+a+(a+a+a+...+a+a)+a+a+a+a+a...

s : a;
a : (t) ('+' t)*;
t : 'a'|'(' a ')';
```

Abstract
Sound & Complexity

Non-Linear Strategies in Music Composition

Argument

This book is based on the application, or the syntagmatic engagement, of a recent philosophical approach in the field of investigations concerning the sound, of its real-time processing, and the elaboration of a complex strategy in music composition. In other words, the project coherently derives from another one, expanded in the linguistic space.

De-construction is a concept (or, rather, an anti-concept) coined by the French philosopher Jacques Derrida with the aim of making ambiguous the semantics of classical, standardized texts of the past and to determine these texts to project new ideology-free meanings. Its aim, therefore, is to extend new from old. Derrida shows that de-construction follows – in its development – two phases: the *shift* (Fr: déplacement) and the *reversal*.

A very simple example could help the understanding of this process; an example taken from musical area, obviously. Beethoven has composed his scores in the well-tempered system (in which the frequencies are given by the powers of $\sqrt{2}$) and by adoption of the great form principle (i.e., a teleological principle concerning the sound organization). Now, the passage to the series of natural harmonics or, to the Chinese pentatonic scale in the field of frequencies - could be equated to a *shift*. And the conversion of the formal strategy from the classical to a repetitive one (the minimalist music of Phlip Glass) could be equated to a *reversal*.

The project supported by this book is made of three levels:

- conceptual / theoretical / strategic
- interactive / concrete / technological
- applicative / informational / pragmatic

The first level consists of: working with generative grammars (*Chomsky*'s context-free grammars), working with fractals, making sound forms using transforms.

The second consists of: introducing the main live electronics technics, and a case study concerning an actual score, namely *Itinéraires à l'intérieur du son I* by Fred Popovici.

The third level consists of: supporting a training process (including master and PhD levels), supporting research and creative approaches (developing new architectures and making new musical compositions).

Theoretical level

Generative grammars

Every specified formal language can be associated to a grammar that entirely describes it. The musical language is not, strictly speaking, a formalized language, because it emerged from an historical process and it has only few features in common with the natural languages. The technical and the stylistical restrictions that influenced, within this evolution, the musical language induced a set of important formal aspects. We can state, today, the musical language of a certain epoch shows several dimensions that could be formalized.

Let be the following simple example of a context-free grammar:

$$G_1 = (\{S, A\}, \{a, b, c\}, \{S \to aAa, A \to aAa \mid bAb \mid c\}, S)$$

It generates, for example:

$$\begin{gathered} S \to aAa \\ aAa \to aaAaa \\ aaAaa \to aabAbaa \\ aabAbaa \to aabaAabaa \\ aabaAabaa \to aababAbabaa \\ aababAbabaa \to aababcbabaa \end{gathered}$$

◇

The use we have in mind for the generative grammars has two aims:

- to ***translate*** from an initial formal language (extracted from classical samples of music) to another one, called destination language (designed for generating new sound forms); *parsing* is the main mechanism used in this approach
- to ***transform*** a sound organism into another ones using different generative decisions inside the same grammar.

Translating with grammars

We provide an example of applying the translation within the limited context of the rhythmical patterns. We extracted the source grammar, G_{M1}, from some excerpts belonging to the late piano sonatas by Mozart (KV 331, KV 332, KV 570). Then we defined the destination grammar, G_{M2}, by defining a new vocabulary and a new set of rules. Afterwards, we used a sample from another sonata (KV 494), we parsed it in G_{M1} and translated it in G_{M2} in order obtain a new rhythmical organism.

Actually we performed the following steps. The rhythmical patterns used to extract G_{M1} are (the beginning of the three sonatas):

```
4    2   4  2  4    2  4  2   |   4  2  4  2  4  2    4  2
3 1  2   4  2  3 1  2  4  2   |   4  2  4  2  4  1 1  4  2


4 2  4 2  4 2  2   4   |   2 4    2   4    2 4  2 4
4 2  4 2  4 2  1 1 4   |   2 2 2  2   3 1  6    6


4 2  4 2  4 2  4 2   | 4       2    4        2    4       2    4       2
4 2  4 2  4 2  4 1 1 | 1 1 1 1 1 1  1 1 1 1  1 1  1 1 1 1 1 1  1 1 1 1 1 1
```

where: 1 stands for semi-quaver, 2 stands for quaver, 3 stands for pointed quaver, 4 stands for crochet, 6 stands for pointed crochet.

From the previous patterns results the grammar:

$$G_{M1} = (\{S, A, B, C\}, \{1, 2, 3, 4, 6\}, P, S)$$

where: P is made of the following productions:
$S \to ASA \,|\, AA$

$A \to BC \,|\, CB \,|\, 6$
$B \to 4 \,|\, 31 \,|\, 22 \,|\, 1111$ $C \to 2 \,|\, 11$

We decided the destination grammar is:

$$G_{M2} = (\{S, A, B, C\}, \{1, 2, 3, 4, 6\}, P, S)$$

unde: P is made of the following productions:
$S \to ASA \,|\, AA$
$A \to BC \,|\, CB \,|\, 6 \,|\, 111111 \,|\, 21111$
$B \to 4 \,|\, 22 \,|\, 31 \,|\, 22 \,|\, 1111$
$C \to 2 \,|\, 11$

We selected the beginning of the KV 332 sonata to be translated. The numerical form for its rhythmical structure is:

```
4 2  4 2  4 2  11 4   |   2 22  2 31  6
```

The resulting translated form is:

```
4 2  22 2  4 2  11 22 22 2  2 4  2 1111 6
```

This result is obviously only statistically similar to the original pattern belonging to the original Mozart's text.

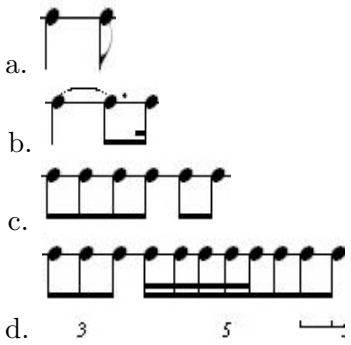


Figure 1.

In the previous example, both the terminals of G_{M1} and of G_{M2} were made of Mozart's usual values of notes. We can extend this vocabulary by putting, instead of them, a set o unusual values. For example, the micro-pattern from the Figure 1a is replaced by the micro-pattern from the Figure

1b. Another example is given by the substitution of the micro-pattern from the Figure 1c with that from the Figure 1d.

Thus, we have new and more complex rhythmical patterns which allow us to go far away from Mozart's flavors.

Generating and transforming a sound organism using grammars

Another use of the generative grammars in our project can be done in the field of generating and transforming sound organisms in order to extend them in larger musical forms. Grammars are used to make discrete transforms on sound organisms generated by discrete mechanisms.

In Figure 2, is shown a sound organism **generated** by a certain grammar. The micro-events, represented by different sized circles, are the products of the action of this grammar beginning from the start symbol, S. The generative tree is symbolised by a triangle, in which is explained how in the node xAy was applied one of the rules of the form $A \to B|C|D$, the result being, in this case, xBy.

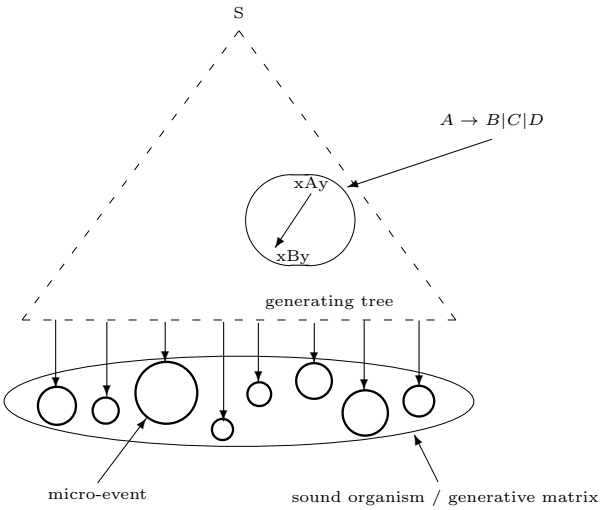


Figure 2. Generating a sound organism.

In Figure 3 is shown the way in which an initial sound organism is transformed by **re**-generation. This apparently means that the micro-events are

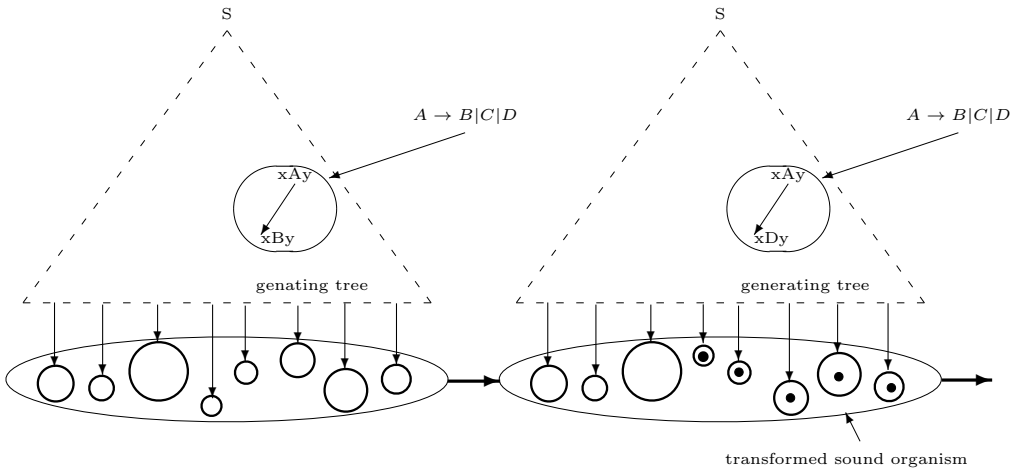


Figure 3. Transforming a sound organism.

subject to a series of distortions. In fact, the process consists of a new generation where in the node xAy we select to apply, instead of $A \to B$ the alternative rule $A \to D$.

Fractals

The appearance, in mathematics, of the theory of fractals represents a tentative to master the real complexity of the worlds by algorithmic procedures taking into account the *apparent complexity*. This theory of fractals preserves the "obsessions" of the Abrahamic culture (i.e., Mosaic, Christian, and Muslim cultures). In this way, this theory states the hierarchical organization involved in the well-known syntagma *"as above, so below"*.

Important definition: *The algorithmic complexity of a given reality is proportional to the length of the shortest description (i.e., of the shortest program) that could generate this reality.*

The apparent complexity is characterized by the following properties:

- simple rules used to generate it
- the generative rules can not be deduced (or at least in an easy way) from the formal analysis of the generated reality.

When between complexity and appearance there is no a legitimate connection, we are compelled to use the *apparent complexity*. It is about a hidden order controlled by the composer generating the appearance of a complexity through which he/she catches the audience's interest. This is a complexity forged in the intimate laboratory of creation, far from the public's perception.

Because fractals are frequently used in contemporary music, the present project is destined to propose the following:

- a new concept regarding the use of fractals in music composition, in respect with an original decisional strategy;
- a new perspective regarding the musical form, freed from the classical teleology;
- a concept of musical form that avoids predictability, followed by the re-injection of order, up to a new equilibrium between order and disorder, to put the premises to the emergent symmetries, and standardized accumulations of sounds. Additionally, we suggest the dissipation of the form, up to the informational "noise".

The concept of *self-similarity* – which is essential to the understanding of this project – makes the fractals the most suitable for the transforms applied to the sound organisms.

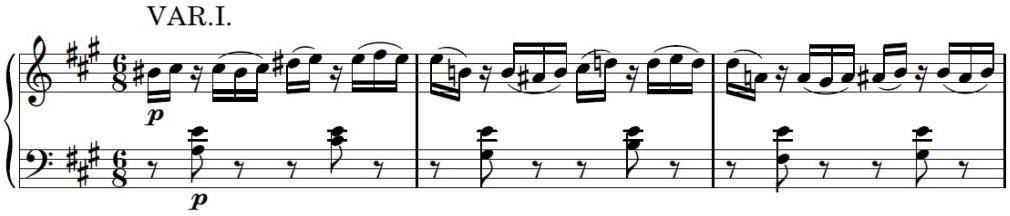


Figure 4. The beginning of the first variation in KV 331 sonata.

Great composers of the past anticipated *as Moliére's Monsieur Joudain*, in their work, the use of fractals in shaping sound forms. For example, Mozart – in the same KV 331 from which we extracted the first fragment used to define G_{M1} – composed the first variation of that tune in the manner of a fractalized melodical envelope (see Figure 4). It's a kind of Cantor' dust.

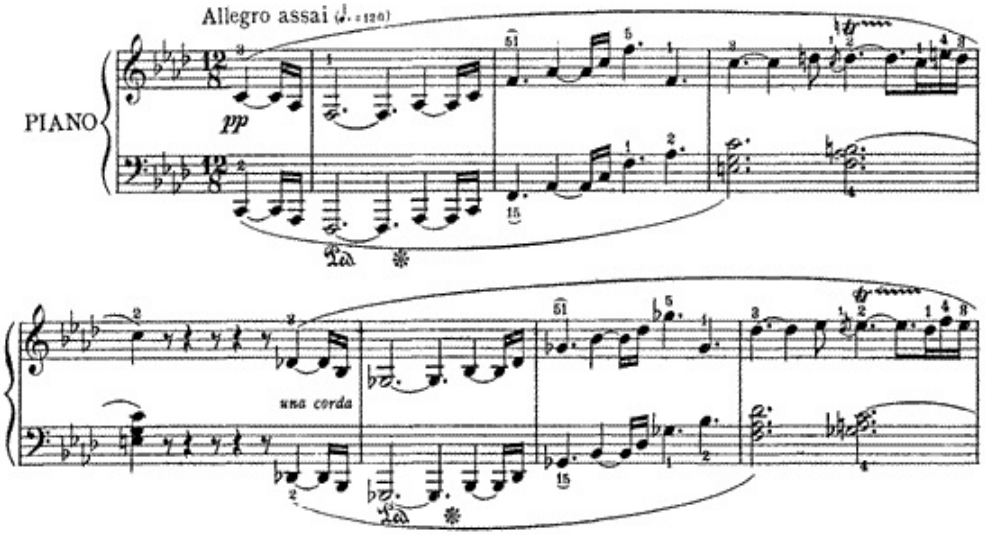


Figure 5. Sonata Appassionata by Beethoven.

More exciting and, perhaps more complex, is what Beethoven made in his famous piano sonata *Appassionata* op. 57. It is a kind of mixture Cantor's dust and Koch's curve (see Figure 5).

In the following we propose two coherent uses of fractals, first in the field of rhythms, second in the field of pitches.

Cantor's dust in the field of rhythms supposes an extraction of duration segment from a sequence of time of 8 seconds. In traditional musical notation it means two tied whole notes at *MM quaver = 60*. In order to make our fractalization we will replace this duration by $6+6$ quaver triplets, because we wish to remove 1/6 from the middle of the time segment. The last value of the second triplet and the first value of the following triplet are replaced by their correspondent rests.

Koch's curve in the field of pitches is illustrated in Figure 6, where the curve is generated using a 45^o angle. From an initial C, represented by the whole segment (see Figure 6a), we go to the first "fracturing" (see Figure

6a). We have now four segments. The first and the last represent the same C, while the second and the third segments are *projected* on the horizontal axis. This means that the same number of sine waves used to generate C are stretched in a shorter time interval. Because this time interval is divided by $\sqrt{2}$ the resulting sound is $F\#$. A similar process is followed up in Figure 6c and Figure 6d. The result is the following sequence of pitches:

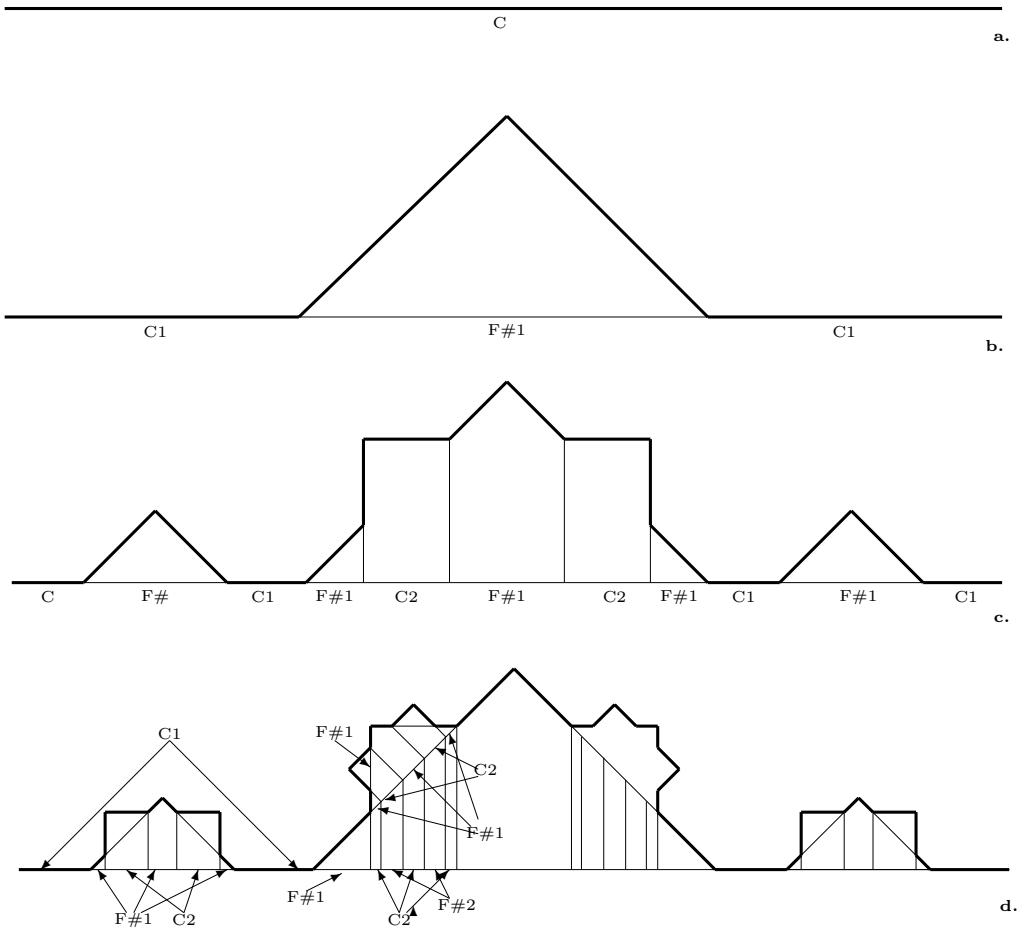


Figure 6. "Fracturing" a C on Koch's curve.

C, F#, C, F#, C1, F#, C1, F#, C1, F#, C1

More different and complex results are expected if we would chose an angle less than 45^o. On the other hand, further complex results are also

expected if we chose a melodical line instead of a simple long sound.

Transforms of forms

The birth of the sound form is a quintessential matter for the present project. Therefore, we give these two definitions:

Definition 1 *A sound organism is a set of micro-events made of different combinations of sound parameters.*

◇

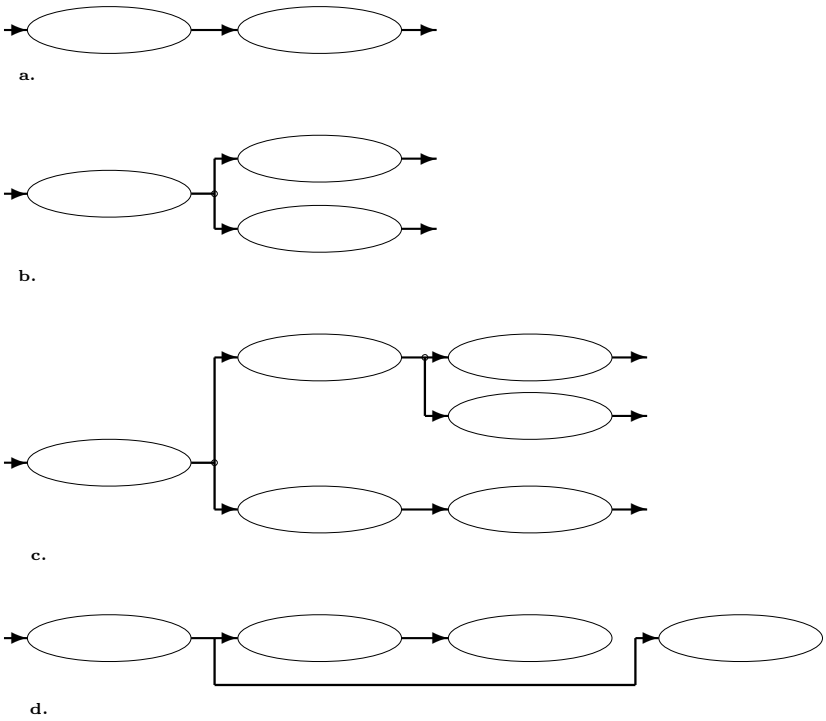


Figure 7. The three ways of growing sound forms.

Definition 2 *A sound form is a set of sound organisms that have direct transformational connections between them.*

◇

There are three ways of growing sound forms: serial connection of sound organisms (see Figure 7a), parallel connection of sound organisms (see Figure 7b), and serial-parallel connection of sound organisms (see Figure 7c). Figure 7d illustrates a forbidden construct.

There are two sources of sound organisms used to grow musical forms: the discrete transforms and the continuous transforms. We previously dealt with the discrete transforms. Now we will present the continuous transforms.

Continuous transforms

The starting point is given by the technical procedure known in electroacoustics as the *reinjection loop*. In a way, it is a metaphor which shows how a sound material is continuously degraded by its reiterated recording. There are two effects applied to a signal, suggested by this metaphor:

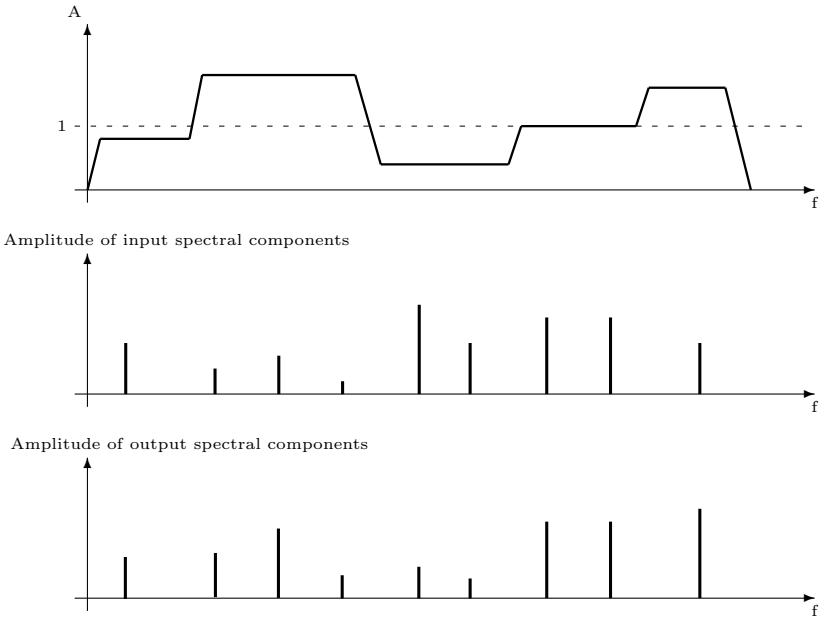


Figure 8. Filter/Amplifier.

- a filter/amplifier function

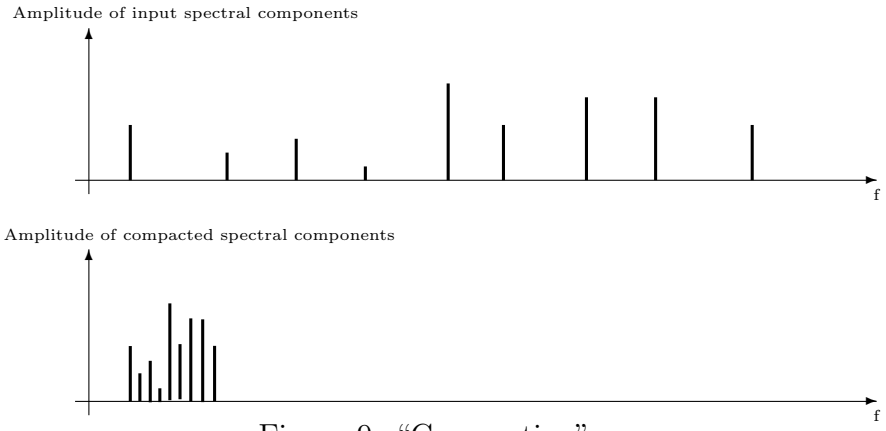


Figure 9. "Compaction".

- a function of adding a "noise" generated by the compaction of the spectral components of the signal.

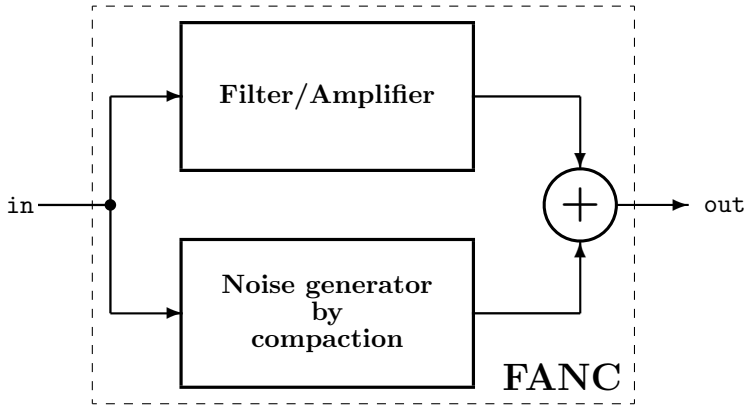


Figure 10. FAZT: **F**ilteer **A**mplifier and **N**oise generated by **C**ompaction.

The filter/amplifier function is described in Figure 8. Unlike a common filter, our filter/amplifier performs both a filtration and an amplification of the input signal.

The second effect is adding, to the filter/amplified signal, a sort of noise. Let us consider a fundamental, f^1, with its set of harmonics. We can generate

a sort of noise by compacting these harmonics between f^1 and its second harmonic, f^2 (see Figure 9). Practically, the two functions described above, are performed by the architecture presented in Figure 10.

Practical level

Case study

In order to verify our point of view, our strategy concerning the appliance of live electronics technics, we have chosen a score by Fred Popovici: *Itinéraires à l'intérieur de son I*. In this piece of music, two sections of the first movement are particularly suitable to the work with live electronics. This score is written for the following ensemble: Solo Flute, *Concertino* Group (flute, clarinet, percussion) and orchtestra.

About making the score

The starting point is given by an information taken from psychoacoustics: the frequency 16.5 Hz is commonly considered as a threshold between pitches and rhythms, in our perception. The high frequencies (pitches) are to be found above 16.5 Hz, the others, the low frequencies (rhythms), are to be found bellow it. Because the instruments of the orchestra are unable to play sounds with less then 33 Hz, the composer has decided that this frequency (the double of 16.5 Hz) will be taken as landmark of the entire work.

Let us consider the following set of frequencies: 33 Hz, 44 Hz, 49.5 Hz, 55.68 Hz, 74.25 Hz. These frequencies represent the fundamentals of a canonical *harmonic sequence*: C – F – G – (C) – A – D – G – [C]. Their frequencies were calculated as follows:

$$\begin{aligned} &33Hz \times 1 && \rightarrow C \\ &33Hz \times 4/3 && \rightarrow F \\ &33Hz \times 3/2 && \rightarrow G \\ &74,25Hz \times 3/4 && \rightarrow A \\ &49,5Hz \times 3/2 && \rightarrow D \end{aligned}$$

Above all these fundamentals we will build their odd-numbered harmonic spectra. Thus results:

$33Hz \times \{1, 3, 5, \ldots (2n-1)\} = \{33Hz, 99Hz, 165Hz, \ldots\},$
$44Hz \times \{(1, 3, 5, \ldots (2n-1)\},$
$49, 5HzHz \times \{1, 3, 5, \ldots (2n-1)\},$
$55, 68Hz \times \{1, 3, 5, \ldots (2n-1)\},$
$74, 25Hz \times \{1, 3, 5, \ldots (2n-1)\},$

On the other hand, we have the spectra of durations (rhythms) by dividing the same fundamentals by the same odd numbers:

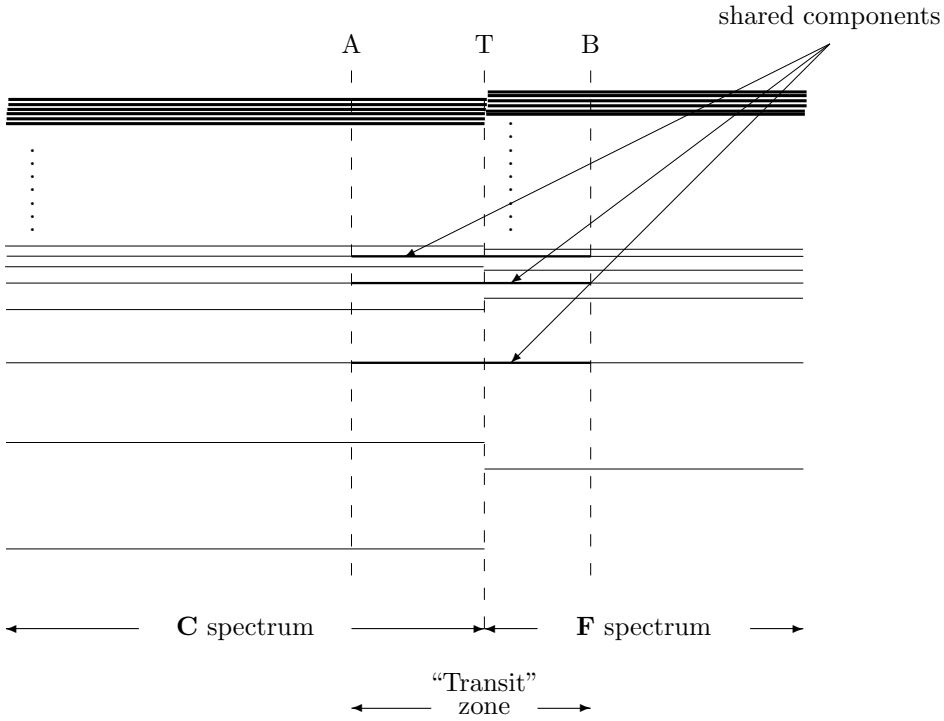


Figure 11. Transition between C and F spectra.

$16, 5Hz/\{1, 3, 5, \ldots (2n-1)\} = \{8, 25Hz, 4, 125Hz, 2, 06Hz, \ldots\}$
$22Hz/\{1, 3, 5, \ldots (2n-1)\}$
$24, 75Hz/\{1, 3, 5, \ldots (2n-1)\}$

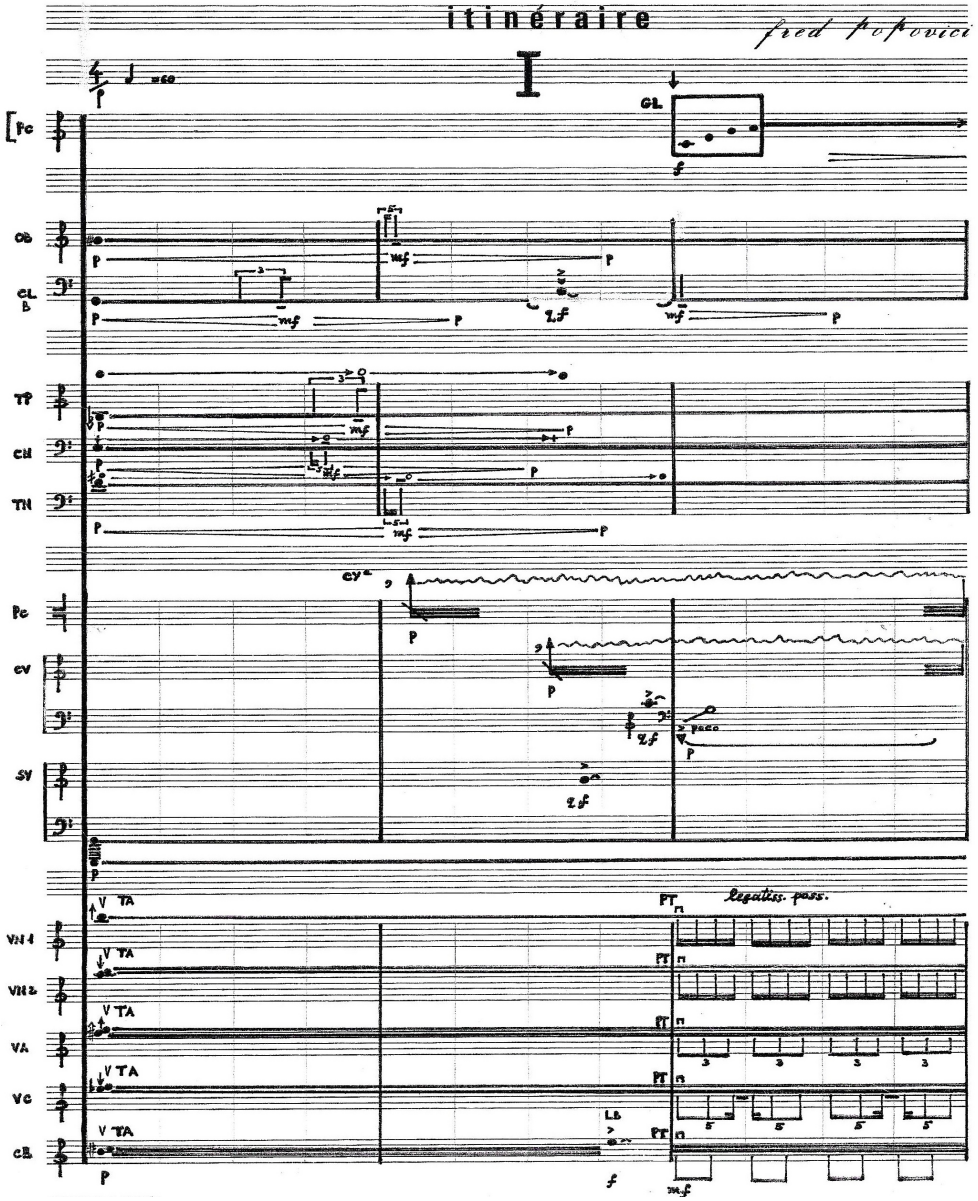


Figure 12. First page of *Itinéraires à l'intérieur du son I.* and the shared components of C and F (see PC., GK).

$27,84Hz/\{1,3,5,\ldots(2n-1)\}$
$37,125Hz/\{1,3,5,\ldots(2n-1)\}$

Composer's strategy in this piece of music requests that all these spectra (of harmonics and durations) might be organized in a chain, observing the above mentioned harmonic sequence. In the same time, the chaining of two fundamentals (with their harmonics) emphasizes a set of shared pitches. *Concertino* group is in charge with the emphasis of the shared pithes.

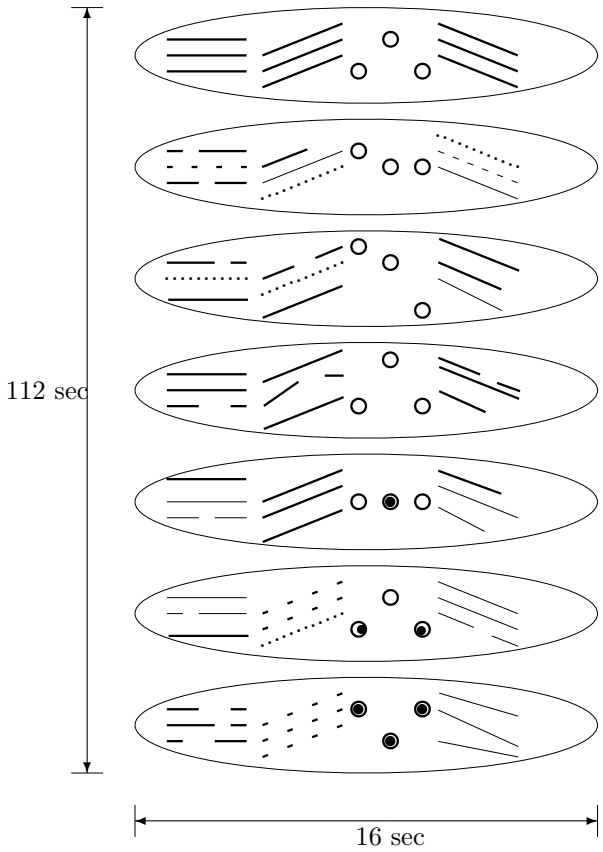


Figure 13. Simulation of the re-injection loop.

Let be, for example, the chain C – F, that gives the intersection $\{SOL\} \cap \{DO\}$. That means, for example, the harmonics 4^0, 12^0, 20^0 of C, are retrieved as the harmonics 3^0, 9^0, 15^0 of F. In Figure 11 is represented

this transition. The Figure 12 reproduces the page of the score *Itinéraires à l'intérieur du son I.*, where the shared pithes of C and F are played by the *Concertino* percussion (Glockenspiel).

The second section from the first movement of the piece submitted to the live electronics process is shaped as follows. There is a sound organism submitted to this process. The sound organism:

- has a duration of about 16 seconds
- is made of the superimposing of three melodical contours. The first one is made of a few harmonics of C, the second one is made of a few harmonics of F, the third is made of a few harmonics of G. The three "melodies" are played by the Solo Flute and the three instruments of the *Concertino* group

During seven 16-second "episodes", i.e., $16 \times 7sec = 112sec$, the initial sound organism is continuously transformed, distorted and dissipated. All the sound parameters are "fractalized". În Figure 13 is presented a re-injection loop, that encompasses the above mentioned transformation processes.

Performing the score

There are two main aspects in performing the score by using live electronics: (1) processing the transition between two spectra, (2) processing the re-injection loop.

The transition between the spectra of C and F is emphasized by the glockenspiel playing the shared components of the two spectra. The instrument is beginning to play few seconds after the occurrence of C and its spectrum (at the moment indicated by A in Figure 11) and continues its playing few seconds after the occurrence, at the moment T, of F and its spectrum (till the moment indicated by B in Figure 11).

The play of the glockenspiel is DELAY-ed between A and T and REVERB-ed between T and B (see Figure 14). Simultaneously, on the shared components – x, y, z – are applied, between A and T, the amplitude envelopes a1, a2, a3, while between T and B the envelopes a4, a4, a6 are applied.

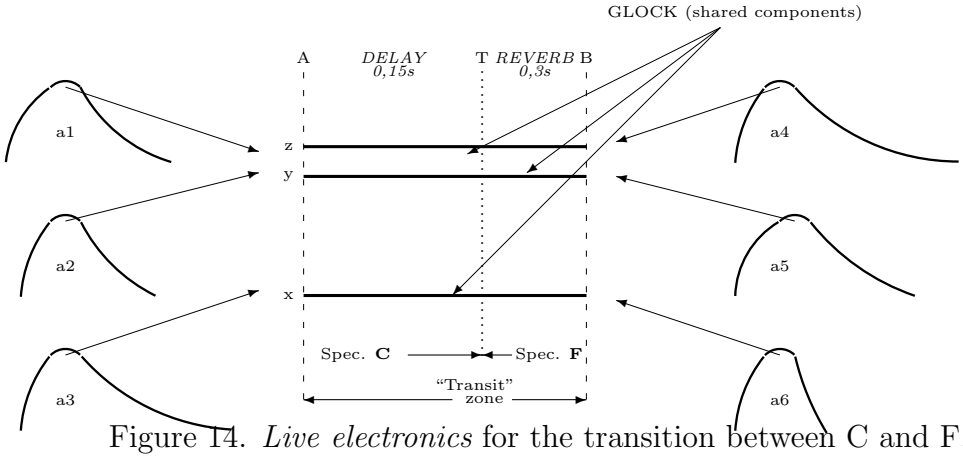


Figure 14. *Live electronics* for the transition between C and F.

Processing the re-injection loop is represented in Figure 15. It contains the 7 episodes of the sound form; above it, the evolution of the gain applied to the "compaction" noise, and bellow it, the evolution of the filter/amplifier profile. In each `OrgSon` there are some micro-events used to determine the changes both in `Gain of the ‘‘compaction" noise` and F/A evolution.

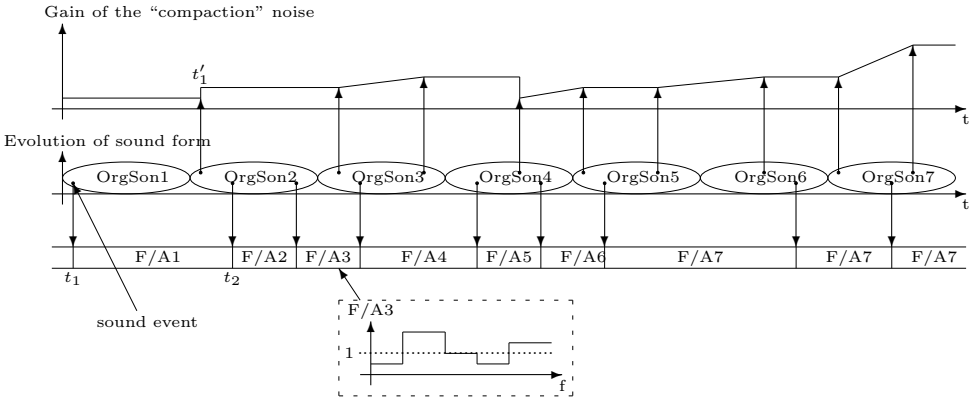


Figure 15. The interaction between the execution of sound form and FANC.

In Figure 16 is shown a page from the score, belonging to this re-injection loop, at the level of the instrumental performance (the sequence of `OgrSon1, ... OrgSon7`).

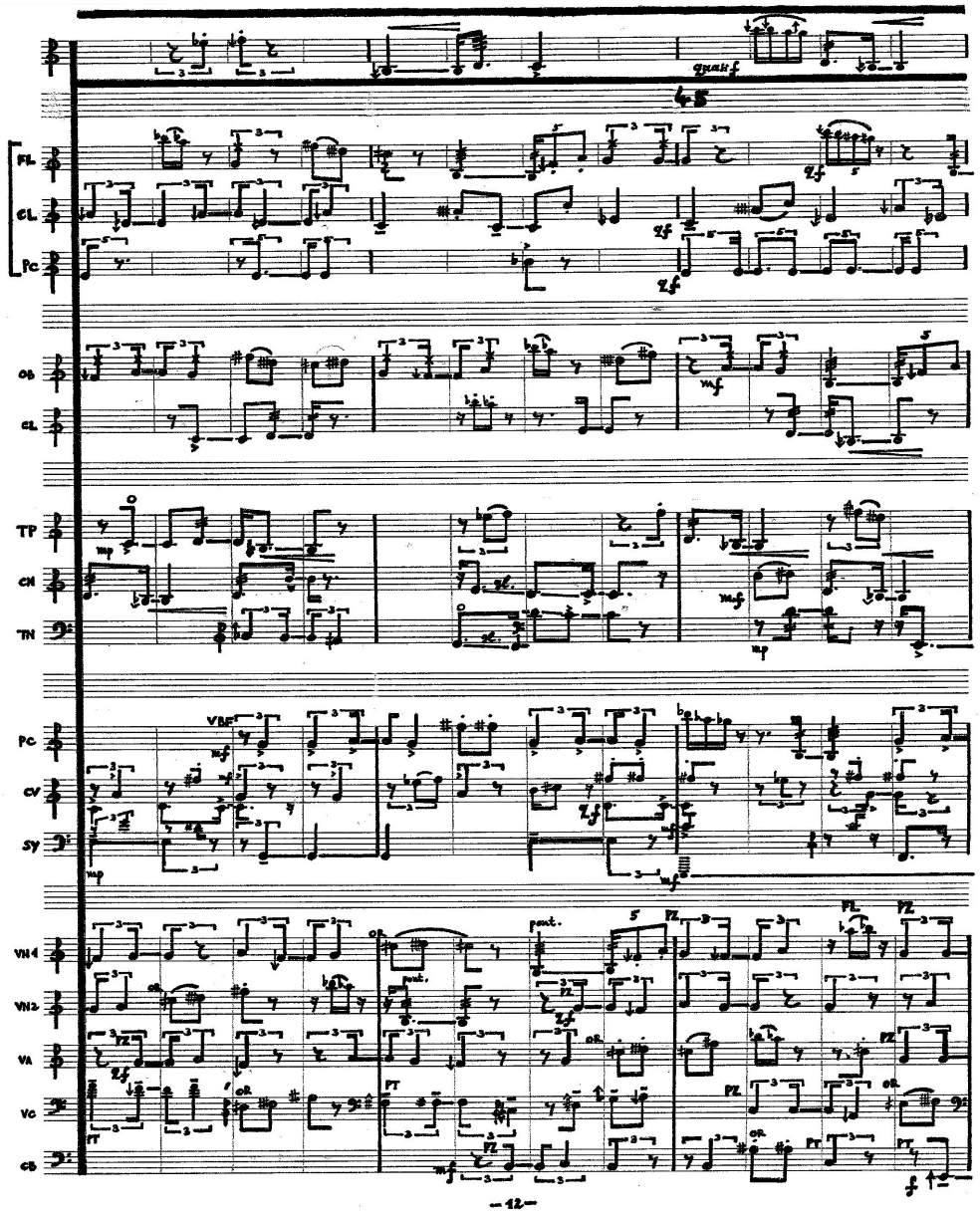


Figure 16. A page from *Itinéraires à l'intérieur du son I.*

Live electronic architecture

The architecture we propose has, in our opinion, the following advantages: (1) improves the efficiency in working with live electronics technics, (2) *ipso facto* a greater assistance offered to the composers in their work, last but not least, (3) easier relations between the creator and his/her performing team.

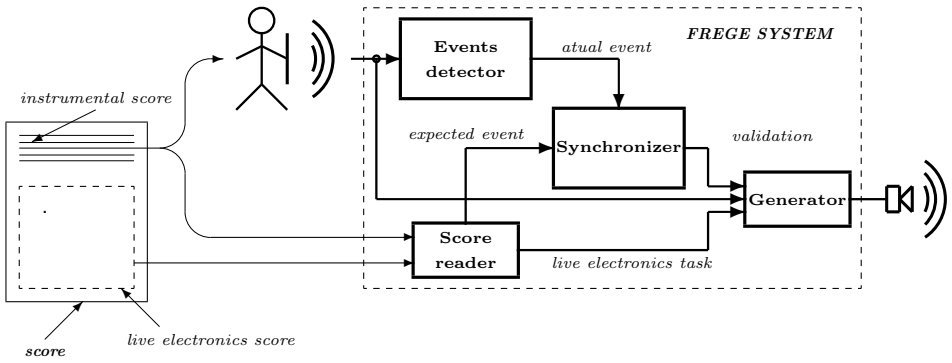


Figure 17. Live electronics FREGE system.

FREGE Architecture presented in Figure 17 is made of:

Score: standard score for instruments plus live electronics score

Live electronics system is made of:

Score reader follows the symbols inscribed in the two scores

Events detector receives the conversion of the symbols of the scores in sound signals and converts them back into symbols

Synchronizer detects the production of sound micro-events established as remarkable in the live electronics score

Generator has three inputs:

- from **Synchronizer**, a signal starting a live electronics transform (it is about *when?*)
- from instrumentalist, the signal that will transformed (it is about *what?*)
- from **score reader** receives the name of the transform to be applied as a symbolic configuration (it is about *how?*)

Bibliografie

[1] Carlos Agon, Gérard Assayag, Joshua Fineberg, Camilo Rueda: *Kant: a Critique of Pure Quantification*, research report IRCAM, 1994.

[2] A. V. Aho, J D. Ullman: *The Theory of Parsing Translation and Compiling*, Prentice-Hall, 1973.

[3] Pierre Boulez, Andrew Gerzsó: "Computers in Music", *Scientific American*, aprilie 1988, vol. 258, no. 4.

[4] Gregory Chaitin: "On the Length of Programs for Computing Binary Sequences", *J. of the ACM*, Oct., 1966.

[5] Gregory Chaitin: "On the Difficulty of Computation", in *IEEE Transactions of Information Theory*, ian. 1970.

[6] Gregory Chaitin: "Algorithmic Information Theory", *IBM J. Res. Develop.*, Iulie, 1977.

[7] Gregory Chaitin: *Algorithmic Information Theory*, Cambridge University Press, 1987.

[8] Gregory Chaitin: *Information, Randomness and Incompletness*, World Scientific,1990.

[9] Gregory Chaitin: *The Limits of Mathematics IV*, IBM Research Report RC 19671, e-print chaodyn/9407009, July 1994.

[10] Gregory Chaitin: *Meta Math! The quest for Omega*, Pantheon Books, 2005.

[11] Gregory Chaitin: "The Limit of Reason", in *Scientific American*, Martie, 2006.

[12] Noam Chomsky, "Three Models for the Description of Languages", *IEEE Trans. on Information Theory*, 2:3 , 1956.

[13] Noam Chomsky: *Syntactic Structures*. Mouton, The Hague, 1957.

[14] Noam Chomsky, "On Certain Formal Properties of Grammars", *Information and Control*, 2:2, 1959.

[15] Noam Chomsky, "Formal Properties of Grammars", *Handbook of Mathematical Psychology*, Wiley, New-York, 1963.

[16] John Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation", *Journal of the Audio Engineering Society*, 21(7):526-534, 1973.

[17] John Chowning, "Frequency modulation synthesis of the singing voice", Max V. Mathews and John R. Pierce, editors, *Current Directions in Computer Music Research*, pages 57-64. MIT Press, Cambridge, 1989.

[18] Ion Creangă, ş. a.: *Introducere algebrică în informatică. Limbaje formale*, Ed. Junimea, 1974.

[19] Jacques Derrida: *De la grammatologie*, Editions de Minuit, 1967.

[20] Jacques Derrida: *l'Ecriture et la différence*. Editions du Seuil, Paris, 1967.

[21] Jacques Derrida: *La dissémination*, Editions du Seuil, Paris, 1972.

[22] Gérard Grisey: "Écrits ou l'invention de la musique spectrale", Guy Lelong and Anne-Marie Réby (ed.), *Répercussions*, Musica Falsa, Paris 2008.

[23] Douglas Hofstadter: *Metamagical Themas: Questing For The Essence Of Mind And Pattern*, Basic Books, 1985.

[24] Andrey Nikolaevich Kolmogorov: "Three Approaches to the Quantitative Definition of Information". *Problems Inform. Transmission*, vol. 1, nr. 1, 1965. pp 17.

[25] Fabien Lévy: "L'écriture musicale à l'ére du numérique", *Culture & recherche* no. 91-92, juillet 2002.

[26] Benoît Mandelbrot: "How Long is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension". *Science* **156** (3775), 1967. pp. 636638.

[27] Benoît Mandelbrot: *The Fractal Geometry of Nature*, Freeman, San Francisco, 1975

[28] Wolfgang A. Mozart: *Sonaten fur Klavier*, Breitkopf, Berlin, 1972.

[29] Angelo Orcalli: *Fenomenologia della musica sperimentale*, Sonus Edizioni, Bolognia, 1993.

[30] Fred Popovici: *Consecinţe ale analizei electroacustice a fenomenului vibratoriu*, Bucuresti, Ed. Muzicală, 1985.

[31] Ilya Prigogine, Isabelle Stengers: *Noua alianţă. Metamorfoza ştiinţei*, Ed. Politică, 1984. (Ediţia originală: Gallimard, 1979)

[32] Ilya Prigogine, Isabelle Stengers: *Order out of Chaos: Man's new dialogue with nature*. Flamingo, 1984.

[33] Larry Solomon: *The Fractal Nature of Music*, posted at: `http://solomonsmusic.net/fracmus.htm`

[34] Ray Solomonoff: "A Preliminary Report on a General Theory of Inductive Inference", *Report V-131, Zator Co.*, Cambridge, Ma. Feb 4, 1960.

[35] Stephen C. Kleene: "General Recursive Functions of Natural Numbers", *Math. Ann.*, 112, 1936.

[36] Karlheinz Stockhausen: *Texte zur Musik*, Band III, Du Mont Edition, 1970.

[37] Gheorghe Ştefan: *Funcţie şi structură în sistemele digitale*, Ed. Academiei Române, 1991.

[38] Gheorghe Ştefan: "Chomsky's Hierarchy & a Loop-Based Taxonomy for Digital Systems", *Romanian Journal of Information Science and Technology* vol. 10, no. 2, 2007.

[39] Gheorghe Ştefan: *Loops & Complexity in DIGITAL SYSTEMS. Lecture Notes on Digital Design in Giga-Gate per Chip Era (work in endless progress)*, posted at: `http://arh.pub.ro/gstefan/0-BOOK.pdf`

[40] Thomas A. Sudkamp: *Languages and Machines. An Introduction to the Theory of Computer Science*, Addison Wesley, 1997.

[41] René Thom: *Modèles mathématiques de la morphogenèse*, Christian Bourgeois, Paris, 1981.

[42] Alan M. Turing: "On computable Numbers with an Application to the Eintscheidungsproblem", *Proc. London Mathematical Society,* 42 (1936), 43 (1937).

[43] John von Neumann: "First Draft of a Report on the EDVAC", reprinted in *IEEE Annals of the History of Computing,* Vol. 5, No. 4, 1993.

[44] Iannis Xenakis: *Formalized Music: Thought and Mathematics in Composition*, Pendragon Press, 1992.

Glosar

A^* : mulţimea tuturor şirurilor formate peste alfabetul finit A, 4

$L(G)$: limbajul generate de gramatica G, 7

$a \overset{*}{\underset{G}{\Rightarrow}} b$: b este dedus din a într-un număr oarecare de paşi în gramatica G, 7

$a \overset{n}{\underset{G}{\Rightarrow}} b$: b este dedus din a în n paşi în gramatica G, 7

automat celular: este o matrice n-dimensională de celule care evoluază sincron în funcţie de starea proprie şi de stările celulelor dintr-o vecinătate constantă., 70

Bach, Johann Sebastian (1685 - 1750): compozitor german, cel mai de seamă reprezentant al barocului muzical., 74

Beethoven, Ludwig van (1770-1827): compozitor german, a dominat tranziţia dintre clasicism şi romantism., 82

Boltzmann, Ludwig Eduard (1844 - 1906): fizician austriac care a contribuit esenţial la dezvoltarea mecanicii statistice., 89

Carnot, Saadi (1796 - 1832): fizician şi inginer francez considerat parintele termodinamicii. A statuat cel de al doilea principiu al termodinamicii referitor la ireversibilitatea termodinamică a unui sistem închis., 89

celulă muzicală: microeveniment sonor., 92

Chaitin, Gregory (n. 1947): fondatorul teoreiei algoritmice acomplexităţii., 66

Chomsky, Noam (n. 1928): este întemeietorul lingvisticii computaţionale, 3, 11

Chowning, John (n. 1934): informatician şi compozitor american, recunoscut prin lucrarile lui despre sinteza sunetului., 103

complexitate algoritmică: dă măsura complexităţii unei entităţi ca fiind proporţională cu lungimea celei mai scurte descrieri asociate acesteia., 67

complexitate aparentă: caracterizează realităţi a căror simplitate este ascunsă., 69

Derrida, Jacques (1930-2004): filosof