VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

FORMÁLNÍ MODELY A JEJICH APLIKACE V HUDBĚ

FORMAL MODELS AND THEIR APPLICATIONS IN MUSIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JOZEF MAKIŠ**
AUTHOR

VEDOUCÍ PRÁCE **prof. RNDr. ALEXANDER MEDUNA, CSc.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Makiš Jozef**
Program: Informační technologie
Název: **Formální modely a jejich aplikace v hudbě**
Formal Models and Their Applications in Music
Kategorie: Teoretická informatika

Zadání:

1. Dle instrukcí vedoucího se seznamte s různými formálními modely, včetně gramatik a automatů.
2. Zaveďte nové verze těchto modelů dle instrukcí vedoucího.
3. Studujte vlastnosti těchto modelů a jejich jazyků dle instrukcí vedoucího.
4. Aplikujte modely z bodu 2 v hudbě, např. pro klasifikaci vybraných hudebních pasáží.
5. Implementujte aplikace navržené v bodě 4.
6. Zhodnoťte dosažené výsledky. Diskutujte další vývoj projektu.

Literatura:

- Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, Volume 1-3, Springer, 1997, ISBN 3-540-60649-1

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Meduna Alexander, prof. RNDr., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 26. října 2021

Abstrakt

Predmetom práce je štúdium rôznych formálnych modelov, ktoré sú aplikovateľné v hudbe. Tie, ktoré sme aplikovali v tejto práci sú gramatiky s rozptýleným kontextom a absolútne nelimitovaný hlboký zásobníkovým automat. Aplikované sú na hudobnú štruktúru, ktorá sa skladá z témy a variácií. Téma je hlavná myšlienka nejakej skladby a variácie sú jej obmena rôznymi spôsobmi. Výsledok je algoritmus, ktorý je založený na preklade LL gramatiky s rozptýleným kontextom a stvárňuje tvorbu variácii.

Abstract

Main subject of this study are different kinds of formal models. We discuss which of them could be applied in music. Those that are applied are scattered context grammars and absolutely unlimited deep pushdown automa. They are applied in music structure as theme and variations. The main topic of a song is the theme, and variations diversify it. Resulting algorithm is based on compilation of a LL scattered context grammar. This algorithm shows how we can generate variations.

Kľúčové slová

hudba, variácie, gramatiky s rozptýleným kontextom, absolútne nelimitovaný hlboký zásobníkový automat

Keywords

music, variations, scattered context grammars, absolutely unlimited deep pushdown automata

Citácia

MAKIŠ, Jozef. *Formální modely a jejich aplikace v hudbě*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. RNDr. Alexander Meduna, CSc.

Formální modely a jejich aplikace v hudbě

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána prof. RNDr. Alexandra Medunu CSc. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Jozef Makiš
8. mája 2022

Poďakovanie

Ďakujem profesorovi Medunovi za jeho cenné rady a pomoc pri vypracovaní tejto práce. Chcel by som sa poďakovať mojim rodičom za obrovskú podporu v spomienke na tatina, ktorý sa nedožil výsledkov tejto práce.

Obsah

Kapitola 1

Úvod

Formálne modely nám vedia sprostredkovať teoretické prostriedky pre popis rôznych praktických aplikácií. Najrozšírenejšou aplikáciou z nich sú programovacie jazyky a dizajn prekladačov týchto jazykov. Ďalšie aplikácie vieme nájsť v biológii, lingvistike alebo umení. Cieľom tejto práce je aplikácia formálnych modelov v umení, konkrétne v hudbe.

Hudobná štruktúra, na ktorú sú aplikované formálne modely v tejto práci sa nazýva téma a variácie. Väčšina moderných i klasických skladieb sa neskladá z čisto nových tónov a úsekov. Množstvo úsekov sa opakuje, ale nie vždy v ich rovnakom znení. Tieto opakovania sú obmieňané rôznymi spôsobmi. Spôsoby opakovania môžu byť zmena rytmu, melódie alebo tóniny. Súčasťou tejto práce je zistiť, ako môžeme popísať tieto štruktúry. Prípadne nájsť riešenie, ktoré ponúka ich tvorbu.

Za hudbu môžeme považovať štruktúru, ktorá sa riadi určitými pravidlami. Napriek tomu, že obsahuje množstvo pravidiel a predpisov, je to prostriedok pre vyjadrenie myšlienky skladateľa. Táto myšlienka odzrkadľuje nálady, prostredie alebo zážitky skladateľa. Takže sa neriadi žiadnymi deterministickými pravidlami. Hlavná myšlienka, ktorá sa nazýva téma je konečná množina tónov, ktoré sú podľa ich dĺžky začlenené do taktov. Obsahom hudby nie je čisto len téma, ale obsahuje aj jej obmieňanie rôznymi hudobnými prostriedkami. Prepojením takýchto štruktúr vzniká celá skladba, ktorá sa skladá z témy a jej variácií. Príkladom takejto skladby môže byť Dvanásť variácií na „Ah vous dirai-je, Maman“. Vzniknutá skladba je štruktúra, ktorá sa skladá z témy a variácie. Cieľom tejto práce je popísanie tejto štruktúry vhodným formálnym modelom. Výsledkom tohto popisu by malo byť stvárnenie týchto štruktúr, ktoré nesmie obmedzovať tvorbu hudby.

V prípade navrhnutého správneho modelu, ktorý by vedel popisovať túto hudobnú štruktúru, je súčasťou práce ho aplikovať. Ako najvhodnejšia aplikácia, ktorá sa dá v hudbe nájsť je jej tvorba. Gramatiky, ktoré sú súčasťou formálnych modelov umožňujú generovať rôzne reťazce. Preto predmetom aplikácie je vytvoriť gramatiku, ktorá tvorí hudobné reťazce, ktoré by zároveň dávali hudobný zmysel. Na vytvorenie takýchto reťazcov je potrebná gramatika, ktorá vie zvládnuť tvorbu týchto reťazcov. Preto je potrebné zaradiť hudobný jazyk do jeho príslušnej triedy v Chomského hierarchie. Zistenie triedy takéhoto jazyka vedie k odhadu gramatiky, ktorá by mohla takýto reťazce generovať.

Text tejto práce je rozdelený do viacerých sekcii. Kapitola 2 sa skladá z dvoch sekcií. Sekcia 2.1 sa zaoberá základnými pojmami v hudbe, ktorá vedie k pochopeniu použitej hudobnej terminologie. Sekcia 2.2 rozoberá jazyky Chomského hierarchie. Ďalej rozoberá, do ktorej kategórie je zaradený jazyk pre popis témy a variácii. V kapitole 3 definujeme gramatiku, ktorá bola použitá pre popis hudobnej štruktúry. Spolu s definíciou nového modelu ukazuje jeho aplikácie pre hudobnú štruktúru. Kapitola 4 sa venuje samotnej implementácii

novej verzie modelu, spusteniu tejto implementácie a ukážkami výsledku tejto implementácie. Posledná kapitola 5 vyhodnocuje výsledky práce a poukazuje na ich praktické použitie. Obsahom tejto kapitoly je aj diskusia k ďalšiemu pokračovaniu tejto práce.

Kapitola 2

Základné pojmy a definície

Táto kapitola oboznamuje so základnými pojmami, ktoré vedú k pochopeniu obsahu tejto bakalárskej práce. Vzhľadom na to, že táto práca je úzko spätá s hudbou a hudobnou teóriou, obsahuje jej základy, ktoré sú prevzaté z [1, 2]. Základy formálnych modelov a ich definície sú prevzaté z [9].

2.1 Hudba

Hudba je súčasťou umenia. Základným stavebným materiálom hudby je tón. Pomocou hudby sa skladateľ snaží vyjadriť nejakú myšlienku. Hudobnú myšlienku je možné zapísať notami pomocou notového zápisu. Cieľom tejto podkapitoly je objasnenie základných pojmov hudby, ako i ukážka hudobnej štruktúry téma a variácie.

2.1.1 Tón

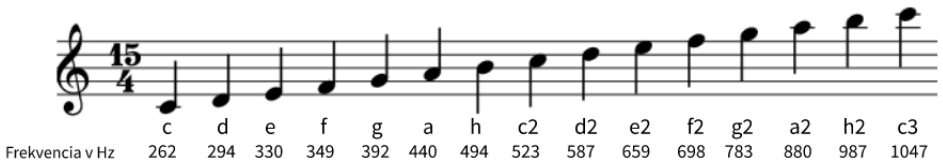
Tón je definovaný, ako zvuk s určitou výškou a vzniká pravidelným chvením nejakého zdroja zvuku. Vzniknuté chvenie sa prenáša do sluchových orgánov zmenou hustoty vzduchu. Skladá sa z rôznych vlastností, ako sú výška, sila, farba a dĺžka. Najdôležitejšie vlastnosti sú výška tónu a jeho dĺžka, ktoré sú predmetom tejto práce.

Dĺžka tónu nie je narozdiel od ostatných vlastností fyzikálnou, ale je závislá od interpreta daného tónu. Čas znejúceho tónu je jeho dĺžkou. Interpret jeho zásahom pri interpretácií vie určiť dĺžku tónu. Interpretom je v našej práci program. Dĺžka tónu je vopred definovaná.

Výška tónu je definovaná ako počet kmitov za jednu sekundu. Spolu s výškou tónu sa zvyšuje počet kmitov za sekundu. Počet kmitov za jednu sekundu sa udáva v Hertzoch.

Tóny sú usporiadané do tónovej sústavy. Tónová sústava je množina tónov, ktoré sa v hudbe vyskytujú. Tvorí ju sedem základných tónov c, d, e, f, g, a, h. Tieto tóny sú výškovo usporiadané od najnižšej po najvyššiu. Po tóne h vždy nasleduje tón c, ktorý má dvojnásobnú frekvenciu oproti tomu pôvodnému. Všetkých osem tónov tvorí oktávu. V tejto práci sú použité dve oktávy jednočiarkovaná oktáva a dvojčiarkovaná oktáva.

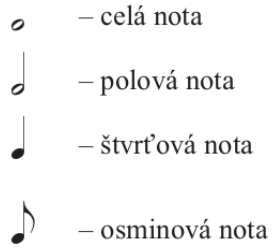
Na obrázku 2.1 sú zobrazené tóny cez dve oktávy, ktoré sú pomenované. K príslušným tónom bola priradená ich frekvencia na základe [15]. Na obrázku môžeme spozorovať, ako frekvencie tónov medzi oktávami súvisia. Jednočiarkovaná oktáva, ktorá sa začína tónom c a končí tónom c2 má polovičnú frekvenciu dvojčiarkovanej oktávy, ktorá sa začína tónom c2 a končí tónom c3. Jednotlivé tóny vieme posunúť o oktávu vyššie vďaka znásobeniu ich frekvencie.



Obr. 2.1: Postupnosť tónov cez dve oktávy s príslušnými frekvenciami.

2.1.2 Nota

Nota je grafická značka, ktorá sa zapisuje do notovej osnovy. Pomocou noty vieme vyjadriť dĺžku tónu a jeho výšku. Ostatné vlastnosti tónu sa zapisujú inými značkami. Zápis nôt do notovej osnovy je potrebný pre reprezentáciu hudobnej myšlienky alebo jej interpretáciu. Noty majú rôzne značenie. Notu tvorí hlavička a nožička. V tejto práci sú použité štyri základné značenia, ako sú celá nota, polová nota, štvrťová nota a osminová nota. Tieto notové značenia sú na obrázku 2.2. Obrázok sme prevzali z [1].

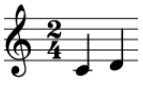


Obr. 2.2: Notové značenia.

2.1.3 Takt

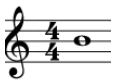
Pod taktom v hudbe rozumieme základnú jednotku pravidelného striedania prízvučných a neprízvučných dôb. Určuje sa v notovom zápise a na začiatku skladby. Značený je dvomi číslami. Vrchné číslo určuje počet dôb v jednom takte. Spodné číslo označuje notu, ktorá trvá jednu dobu.

Na obrázku 2.3 je dvojštvrťový takt, ktorý obsahuje dve štvrťové noty c, d. Dvojka značí to, že v jednom takte sa môžu maximálne vyskytovať dve doby. Štvorka označuje trvanie štvrťovej noty, ktorá trvá jednu dobu. V tomto takte sa nemôže napríklad vyskytovať celá nota, ktorá trvá štyri doby. Tá v tomto prípade presahuje dĺžku jedného taktu.



Obr. 2.3: Dvojštvrťový takt.

Obrázok 2.4 už obsahuje štvorštvrťový takt, v ktorom sa už môže vyskytovať celá nota, ktorá trvá štyri doby.



Obr. 2.4: Štvorštvrťový takt.

2.1.4 Notový zápis

Notový zápis sa skladá z piatich čiar a štyroch medzier. Začína sa kľúčom, ktorý slúži pre relatívne určenie výšky skladby. Existujú rôzne kľúče. V tejto práci sa vyskytuje jeden a tým je husľový kľúč. Husľovým kľúčom sa zapisujú tóny stredné a vysoké.

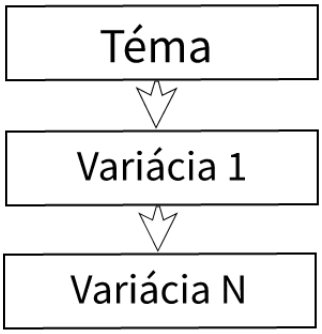
Po husľovom kľúči sa určí tónina. Tónina je množina tónov, ktorá je obsiahnutá v melódií. Každá tónina prislúcha nejakej stupnici. Stupnicu tvorí osem základných tónov, ktoré sú výškovo usporiadané. Stupnica sa začína tónom, ktorý patrí názvu stupnici. C durová stupnica sa začína tónom c. V tejto práci sa pracuje čisto s notami z C durovej stupnice a výsledky práce sú v C durovej tónine.

Zapísaný kľúč a tóninu, v ktorej je melódia zapísaná nasleduje určenie taktového predpisu. Posledná časť notového zápisu je samotná melódia, ktorá je zapísaná notami.

V prípade, že k zápisu nôt do notovej osnovy nestačí počet čiar, potom sa pridávajú pomocné čiary. Príklad pomocnej čiary je pre notu c je na obrázku 2.1.

2.1.5 Téma a variacie

Téma a variácie sú bežné hudobné prostriedky, ktoré tvoria hudobnú štruktúru. Táto hudobná štruktúra je používaná vo veľkej miere v klasickej hudbe. Štruktúra je založená na téme, ktorá sa vyskytuje na začiatku hudobnej myšlienky. Potom, ako odznie téma prichádzajú na rad variácie. Variácia je obmena témy rôznymi spôsobmi. Tento proces je opakovaný niekoľkokrát. Ukončenie tohto procesu záleží na skladateľovi. Z vytvorenej variácie je stále možné vyzistiť originálnu tému, z ktorej bola variácia vytvorená. Proces tvorenia tejto hudobnej štruktúry je zobrazený na obrázku 2.5.



Obr. 2.5: Schéma aplikovania variácií.

Tvorca hudobnej myšlienky môže k variovaniu témy využiť rôzne hudobné prostriedky. Medzi tieto prostriedky patrí zmena melódie. Melódia je hlavný výrazový prvok hudby, ktorý umožňuje vnímanie a pochopenie hudby. Melódia môže ovplyvniť aj náladu hudby.

Variovanie melódie je možné vďaka pridávaniu tónov, odoberaním tónov alebo invertovaním melódie. Počas barokového obdobia sa stali populárne rôzne zdobenia tónov, ktoré môžu takisto variovať tému.

Ďalším spôsobom, ako vytvoriť variáciu témy je zmena rytmu melódie. Zmena rytmu melódie je aplikovaná na tému skladby. Vykonáva sa či už nad jednotlivými notami alebo nad celým taktom. Jednotlivým notám sa môže predĺžiť ich dĺžka alebo aj skrátiť niekoľkonásobne.

Medzi hudobné elementy, ktoré sa dajú zmeniť patrí aj harmónia a tonalita skladby. Cieľom tejto techniky je zmeniť tóninu hudby. V prípade, že skladba je písaná v tónine, ktorá je veselá, potom sa zmení na smutnú. Toto sa dá aplikovať aj opačne a zmeniť smutnú tóninu na veselú. Veselé tóniny sa nazývajú durové a smutné sú molové. Samozrejme, že zmena tóniny je možná v rámci durových alebo molových tónin.

Vytvorenie ďalšej variácie je možné aj zmenou taktového predpisu. Zmenou taktového predpisu prepíšeme tému napríklad z dvojštvrťového taktu na trojštvrťový takt. Spôsobí to zmenu usporiadania nôt v takte. Vďaka tomu sa do jedného taktu zmestí väčší počet nôt, ktoré sú dĺžkovo kratšie.

2.1.6 Bežné druhy variácií

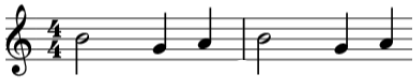
Predmetom rôznych prác je analýza hudby jej častí. Jednou z nich je práca [17], ktorá sa zaoberá vyhľadávaním hudobnej kategórie v hudbe a relevantnosťou výsledku. Zistenie kategórie polyfónnej symbolickej hudby je založené na základe sémantického učenia konceptu hudby. Aby tento systém vzal do úvahy globálne, ale i lokálne vlastnosti hudobných objektov, tak prezentuje prístup k modelovaniu hudobného objektu na základe bázových segmentov.

Modelovanie hudobného objektu sa skladá z viacerých častí. Prvá z nich je získanie potenciálne významných segmentov pre každý hudobný objekt. Ďalším krokom je pre každý významný segment hudobného objektu vybrať prvky lokálnej reprezentácie. Potom sú všetky významné segmenty vyberané na základe ich dôležitosti. Na základe dôležitosti sú vyberané dôležité objekty z tých objektov, ktoré boli označené ako potenciálne dôležité. Posledným krokom je pre každý hudobný objekt získať reprezentácie globálnej vlastnosti.

V hudbe motív, ktorý je výrazne sa opakujúci segment nôt sa používa na zloženie časti alebo celej témy. Opakovanie tohto motívu môže mať nejaké variácie, ktoré nemusia byť vždy presnou kópiou hudobného objektu. Tieto opakujúce sa vzory vo variáciách sa nazývajú motivické opakujúce sa vzory. Tieto motivicky sa opakujúce vzory môžu byť potenciálne dôležité segmenty pre charakterizovanie melódie hudobného objektu.

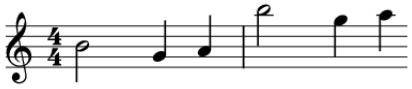
Preto v tejto práci bolo použitých šesť bežných druhov motívových variácií podobne ako v [17]. Tieto motívové variácie sú:

- Opakovanie je presne kopírovanie nôt do ďalšieho taktu ako je na obrázku 2.6.



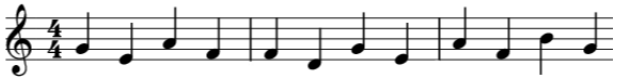
Obr. 2.6: Obrázok opakovania taktu.

- V transpozícii sa motív prenesie na ďalšiu úroveň kmitočtu. Napríklad nota h má kmitočet 494 Hz, takže na ďalšej úrovni bude mať dvojnásobný kmitočet 987 Hz. Príklad tohto posunu je na obrázku 2.7.



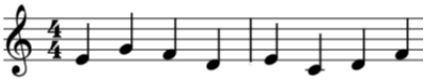
Obr. 2.7: Obrázok transpozície taktu.

- Postupnosť v tomto prípade znamená prenesenie motívu konštantne o jeden tón vyššie alebo o jeden tón nižšie. Na obrázku 2.8 je motív v druhom takte prenesený o jeden tón nižšie, a v treťom takte je prenesený o jeden tón vyššie oproti prvému taktu.



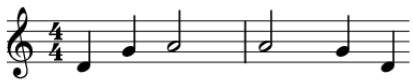
Obr. 2.8: Obrázok postupného zvyšovania a znižovania pôvodného taktu.

- V protipohybe sa zoberú intervaly z prvého taktu, ktorý je v tomto prípade motív. Tieto vzdialenosti v našom prípade zbierame na základe prvého tónu témy. Potom v ďalšom takte sa tieto vzdialenosti tónov invertujú. V prvom takte na obrázku 2.9 sú vzdialenosti medzi tónmi 2, 1, -1. V druhom takte sú vzdialenosti medzi tónmi -2, -1 a 1.



Obr. 2.9: Obrázok protipohybu pôvodného taktu.

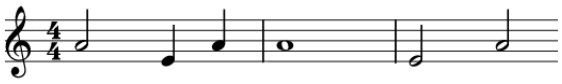
- Retrogradácia v hudbe je zopakovanie nôt z pôvodného motívu v opačnom poradí. Príkladom môže byť motív v prvom takte na obrázku 2.10. Poradie tónov je d, g, e tak jeho variácia v druhom takte je e, g d.



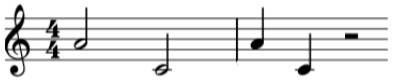
Obr. 2.10: Obrázok retrogradácie prvého taktu.

- V prípade variovania témy predlžovaním alebo znižovaním dĺžky nôt je motív opakovaný. Variovanie motívu prebieha na obrázku 2.11 zdvojnásobením pôvodnej dĺžky noty. Keďže je stále potrebné dodržať rytmus taktu, tak sa museli posledné dve noty presunúť do extra taktu. Z taktového predpisu jeden takt môže obsahovať 4 doby. Po zdvojnásobení tejto dĺžky je potrebné rozložiť 8 dôb do dvoch taktov.

Na obrázku 2.12 je dvojnásobné zníženie dĺžky nôt. Keďže takt trvá 4 doby, potom jeho polovičné zníženie trvá 2 doby. Pre dodržanie tohto taktového predpisu je k dvom dobám pridaná pomlčka, ktorá trvá 2 doby. Pomlčky nie sú súčasťou práce, ale v tomto prípade sú použité pre korektný zápis taktu. Pomlčky sú rozšírené v hudbe. My ich nepoužívame, pretože nami použité variácie sa zaoberajú prácou s notami.



Obr. 2.11: Predĺženie dĺžky prvého taktu.



Obr. 2.12: Skrátenie dĺžky prvého taktu.

2.2 Formálne modely

Zavedenie formálnych modelov bolo vyžiadané čisto matematickým prístupom k jazykom. Veľká časť týchto modelov je založená na prepisujúcich systémoch. Prepisujúce systémy sa skladajú z pravidiel, ktoré opakovane menia poradie symbolov v reťazcoch. Klasifikujeme ich do dvoch základných kategórií. Generatívne modely, ktoré nazývame gramatikami definujú reťazce svojich jazykov. Tieto reťazce sú vygenerované pravidlami prepisovacieho systému z počiatočného symbolu. Prijímajúce modely, ktoré nazývame automatmi definujú reťazce svojich jazykov pomocou prepisovacieho procesu. Tento proces začína z týchto reťazcov a končí prvkom z predom danej konečnej množiny reťazcov.

2.2.1 Množina

Skupinu elementov, ktoré sú prevzaté z nejakého vopred dohodnutého prostredia nazývame množinou Σ. Element a patrí množine M vtedy, ak sa v nej nachádza. Zapisujeme to ako $a \in \Sigma$. V prípade, že sa v množine nenachádza zapíšeme to ako $a \notin \Sigma$. Ak počet prvkov v množine vieme spočítať, potom o množine hovoríme, že je konečná. Ak prvky nespočítame, potom je množina nekonečná. Konečná množina, ktorá neobsahuje žiadne prvky je prázdna množina. Prázdnu množinu označujeme ako $\emptyset$.

Konečnú množinu Σ môžeme špecifikovať vypísaním jej elementov. Obsah množiny zapíšeme ako $\Sigma = \{a_1, a_2, a_3, ..., a_n\}$, kde prvky $a_1, a_2, a_3, ..., a_n$ patria množine Σ.

V práci s množinami využívame rôzne operácie ako sú zjednotenie, prienik a rozdiel. Zjednotenie dvoch konečných množín Σ a Ω definujeme ako $\Sigma \cup \Omega = \{a | a \in \Sigma \, alebo \, a \in \Omega\}$, ich prienik $\Sigma \cap \Omega = \{a | a \in \Sigma \, zároveň \, a \in \Omega\}$, a rozdiel $\Sigma \setminus \Omega = \{a | a \in \Sigma \, zároveň \, a \notin \Omega\}$.

2.2.2 Relácia

Majme dva objekty x a y. Pod označením (x, y) rozumieme usporiadanú dvojicu objektov x,y v tomto poradí. Majme dve množiny X a Y. Kartézsky súčin X,Y značíme ako $X \times Y$. Definujeme ho ako $X \times Y = \{(x, y) | x \in X \, a \, y \in Y\}$. Na základe týchto znalostí môžeme

definovať reláciu. Reláciou r, ktorá je od X po Y rozumieme hociakú podmnožinu $X \times Y$. Pod značením $r \subseteq X \times Y$ rozumieme nami definovanú reláciu.

2.2.3 Reťazec

Konečná množina, ktorej prvky sú symboly nazývame abecedou Σ. Reťazcom nad abecedou Σ nazývame konečnú postupnosť symbolov abecedy Σ. Reťazec, ktorý neobsahuje symboly sa nazýva prazdny reťazec a označujeme ho ako ε. Σ^* označuje množinu všetkých možných reťazcov nad abecedou Σ. Σ^+ označuje $\Sigma^* \setminus \{\varepsilon\}$. Pre $x \in \Sigma^*$ označenie $|x|$ vyjadruje počet symbolov v reťazci x. Podmnožina L, pre ktorú platí $L \subseteq \Sigma^*$ je formálnym jazykom nad Σ. V prípade, že L je konečná množina reťazcov, potom je konečným jazykom. Naopak, ak L je nekonečná množina reťazcov, potom je nekonečným jazykom. Rodina jazykov je množina, ktorej prvkami sú jazyky.

2.2.4 Gramatiky

Podsekcia gramatik má za cieľ zadefinovať gramatiky, ktoré sa vyskytujú v Chomského hierarchii. Gramatiky umožňujú generovať jazyky, ktoré zapadajú do rôznych časti Chomského hierarchie. Postupným zistením, do ktorej rodiny jazykov patrí hudobný jazyk sa zameriame na jednu skupinu gramatik. Tá gramatika by vo výsledku generovala hudobné jazyky.

Definícia 2.2.1 *Frázovú gramatiku definujeme ako štvoricu $G = (N, T, P, S)$, ktorá sa skladá z*

- *abecedy neterminálov N*
- *abecedy terminálov T, kde platí $N \cap T = \emptyset$*
- *konečnej relácie P, ktorá je od $\{N \cup T\}^* N \{N \cup T\}^*$ po $\{N \cup T\}^*$*
- *počiatočný symbol S pre ktorý platí $S \in N$*

Množina $V = N \cup T$ vyjadruje kompletnú abecedu gramatiky G. Prepisujúce pravidlá sa skladajú z dvojíc $(u, v) \in P$, ktoré označujeme ako $u \to v$. Mazacie pravidlo označujeme ako $u \to v, kde\ v = \varepsilon$. Frázové gramatiky generujú rekurzívne spočítatelné jazyky. Rekurzívne spočítatelné jazyky vedia prijať Turingové stroje.

Definícia 2.2.2 *Kontextová gramatika je frázová gramatika $G = (N, T, P, S)$, ktorá sa skladá z prepisovacích pravidiel $u \to v$ vo forme*

$$u = x_1 A x_2, v = x_1 y x_2.$$

Platí, že $A \in N$, $x_1, x_2 \in V^*$. Ďalej $y \in V^+$. Potom hovoríme o kontextovej gramatike. Kontextové gramatiky generujú kontextové jazyky. Kontextové jazyky vieme prijať linearne ohraničenými automatmi.

Definícia 2.2.3 *Bezkontextová gramatika je frázová gramatika $G = (N, T, P, S)$, ktorá sa skladá z prepisovacích pravidiel vo forme*

$$A \to x.$$

Tu platí $A \in N$, $x \in V^*$. Bezkontextové gramatiky generujú bezkontextové jazyky. Bezkontextové jazyky vedia prijať zásobníkové automaty.

Definícia 2.2.4 *Lineárna gramatika je frázová gramatika $G = (N, T, P, S)$, ktorej všetký pravidla sú v tvare*

$$A \to xBy, alebo\, A \to x.$$

Kde A,B $\in N$ a x,y $\in T^*$. Lineárny jazyk je generovaný linearnou gramatikou.

Definícia 2.2.5 *Regulárna gramatika je frázová gramatika $G = (N, T, P, S)$, ktorá sa môže skladať z pravidiel vo forme*

$$A \to aB, alebo\, A \to a.$$

Pravidla sa skladajú z A,B $\in N$, a $\in T$. Regulárne gramatiky generujú regulárne jazyky. Pre prijatie regulárnych jazykov používame konečné automaty.

Definícia 2.2.6 *Rodina regulárnych jazykov môže byť popísaná pravo-lineárnymi gramatikami, ktoré sú definované nasledovne. Pravo-lineárna gramatika je frázová gramatika $G = (N, T, P, S)$, ktorá má všetky pravidla v tvare*

$$A \to aB, alebo\, A \to a.$$

Platí, že A,B $\in N$ a x,y $\in T^*$. Pravo-lineárny jazyk je generovaný pravo-lineárnou gramatikou.

2.2.5 Automaty

Podsekcia automaty definuje základné prostriedky, ktoré sa používajú pre rozpoznanie reťazcov. Tieto reťazce patria rôznym jazykom. V prípade, že daný reťazec nepatrí jazyku, ktoré prijíma konkrétny automat, tak je tento reťazec odmietnuty. Ich využitie leží v prijatí hudobného jazyka.

Definícia 2.2.7 *Konečný automat je definovaný ako pätica $M = (Q, \Sigma, R, s, F)$, ktorá sa skladá z*

- *konečnej množiny stavov Q,*
- *vstupnej abecedy Σ,*
- *množiny pravidiel alebo prechodov R, ktoré sú konečnou reláciou $R \subseteq Q \times \Sigma^* \times Q$,*
- *počiatočného stavu $s \in Q$,*
- *množiny konečných stavov $F \subseteq Q$.*

Pravidla sú vo forme $py \to q \in R$ na rozdiel od $(p, y, q) \in R$. Ak $y \neq \epsilon$, potom hovoríme, že M je bez epsilon prechodov. Konfiguráciu automatu M tvorí hociktorý reťazec z $Q\,\Sigma^*$. Značenie $\vdash_M$ značí reláciu pohybu, ktorá je definovaná ponad $Q\,\Sigma^*$ ako

$$pyx \vdash_M qx.$$

Definícia platí ak a jedine ak $pyx, qx \in Q\,\Sigma^*$, a zároveň $py \to q \in R$. $L(M)$ značí jazyk automatu M, ktorý je definovaný

$$L(M) = \{w \in \Sigma^* \mid sw \vdash_M^* f, f \in F\}.$$

Definícia 2.2.8 *Zásobníkový automat je konečný automat obohatený o zásobník. Zásobníkový automat je teda definovaný sedmicou $M = (Q, \Sigma, \Gamma, R, s, S, F)$, ktorá sa skladá z*

- *konečnej množiny stavov* Q,
- *vstupnej abecedy* Σ,
- *zásobnikovej abecedy* Γ,
- *pravidiel alebo prechodov* R, *ktoré sú konečnou reláciou* $R \subseteq \Gamma^* \times Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^* \times Q$,
- *počiatočného stavu* $s \in Q$,
- *počiatočného symbolu zásobníka* S,
- *množiny konečných stavov* $F \subseteq Q$.

Pravidlá sú vo forme $\gamma pa \to wq$ *na rozdiel od* $(\gamma, p, a, w, q) \in R$. *Konfiguráciu automatu* M *tvorí hociktorý reťazec z* $\Gamma^* Q \Sigma^*$. *Značenie* $\vdash_M$ *značí reláciu pohybu, ktorá je definovaná ponad* $\Gamma^* Q \Sigma^*$ *ako*

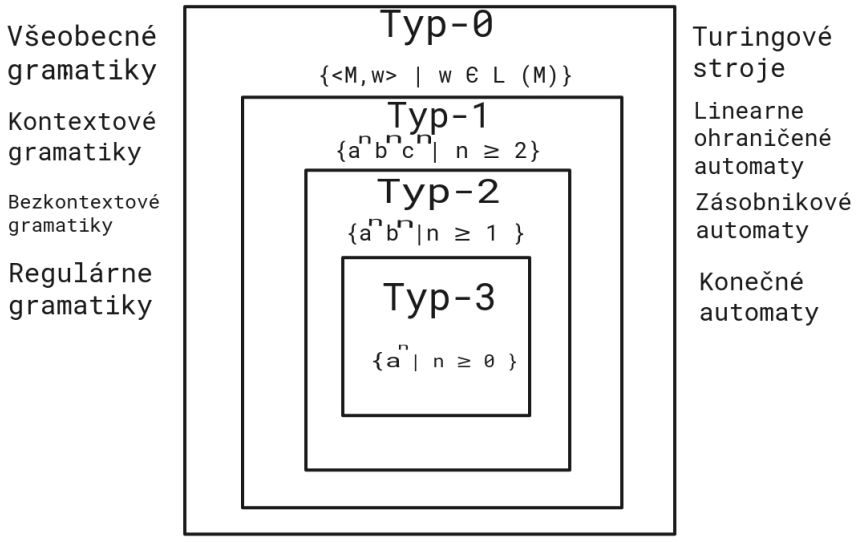
$$x\gamma pay \vdash_M xwqy.$$

Definícia platí ak a jedine ak $x\gamma pay, xwqy \in \Gamma^* Q \Sigma^*$, *a zároveň* $\gamma pa \to wq \in R$. *Existujú rôzne spôsoby ako prijať jazyk zásobníkovým automatom. Súvisiaci je ale jeden spôsob a to s prázdnym zásobníkom.* $L_e(M)$ *značí jazyk automatu* M, *ktorý prijíma jazyk prázdnym zásobníkom nasledovne*

$$L_e(M) = \{w \in \Sigma^* \mid Ssw \vdash_M^* q, q \in Q\}.$$

2.2.6 Chomského hierarchia

Frázové, kontextové, bezkontextové a regulárne gramatiky tvoria jazyky typ-0, typ-1, typ-2 a typ-3. Rodina jazykov generovaná pravo-lineárnymi gramatikami sú si rovné s rodinou jazykov, ktoré sú generované regulárnymi gramatikami. Značenia RVJ, KJ, BJ, LINJ, REJ sú použité pre rodiny jazykov generovaných všeobecnými, kontextovými, bezkontextovými, lineárnymi a regulárnymi gramatikami. RLINJ je označenie pre rodinu jazykov generovanú pravo-lineárnymi gramatikami. Pre tieto rodiny jazykov platí inklúzia $REJ = PLINJ \subset LINJ \subset BJ \subset KJ \subset RVJ$. Popis tejto hierarchie a jej prepojenie s hudbou je prevzaté z [14, 7].



Obr. 2.13: Chomského hierarchia slabej generatívnej kapacity.

Obrázok sme prevzali z [7].

Typ-3 (Regulárne jazyky)

Regulárne jazyky sa používajú napríklad pre zápis regulárnych výrazov. Tie sa využívajú v rôznych UNIXových nástrojoch. Tieto jazyky sú výpočtovo nenáročné, deterministicky spracovateľné v lineárnom čase a podporujú rôzne matematické operácie, ako sú napríklad zjednotenie alebo konkatenácia. Ich najväčšia nevýhoda oproti silnejším formálnym modelom je ich limitovaná generatívna kapacita.

Regulárne gramatiky sú taktiež veľmi limitujúce v tom, že nedokážu vytvoriť viacúrovňovú stromovú štruktúru, ktorou hudobná štruktúra určite je. Je to kvôli obmedzeniu gramatiky, u ktorej musí byť jeden neterminal na pravej strane produkčných pravidiel. Spracovanie hudobnej štruktúry si vyžaduje minimálne možnosť spracovať vnorené závislosti, ktoré sa v hudbe vyskytujú. Túto možnosť regulárne gramatiky neponúkajú. Príkladom stromovej štruktúry môže byť variácia na obrázku 2.10.

Spočiatku sa ponúkala ako vhodné riešenie pravo-lineárna gramatika definovaná v 2.2.6. Tieto gramatiky spadajú na základe Chomského hierarchie do tejto kategórie. Pravidlami tejto gramatiky by sme mohli popísať tvorbu jednotlivých tónov. Majme pravidlo pravolineárnej gramatiky $A \rightarrow aB$. Ak vieme dosadiť za a nejaký tón, potom vieme deriváciami tvoriť reťazce tónov tvoriace hudbu. Napriek tomu, že ponúkajú túto možnosť, nie je ich možné použiť pre popis hudobnej štruktúry alebo tvorby hudobných pasáži. Variácie, ktoré sú rozoberané vyššie maju medzi sebou rôzne križiace a vnorené závislosti. Tieto vlastnosti pripisujeme jazykom typ-2 a typ-1 a nie je možné ich popísať regulárnymi a pravo-lineárnymi gramatikami. Uplatnenie týchto gramatik by bolo možné v prípade využitia derivácií pre tvorenie hudby. Samozrejme, že derivačné kroky by museli byť riadené nejakým spôsobom. Jedným z nich by bola pravdepodobnosť. Pri taktomto postupe nie je možné zaručiť rozumný výsledok. Výsledok takéhoto postupu by nemusel byť vhodný na počúvanie, pretože by sa skladal z náhodného zhluku tónov.

Markovovské procesy, ktoré boli charakterizované Chomskym patria medzi typ-3 a nepreukázali schopnosť spracovať frázovú štruktúru. Tieto limitácie nesúvisia s problematikou stochastických procesov a deterministickych procesov. Limitácie sa týkajú produkčných pravidiel. Markovovské procesy sa riadia lineárnymi krokovými pravidlami narozdiel od bezkontextových gramatík. Bezkontextové gramatiky umožňujú vytvoriť reťazec neterminálov naraz, pomocou produkčných pravidiel. Na základe týchto skutočností vieme, že regulárne gramatiky neponúkajú vhodne riešenie k popisu hudobnej štruktúry.

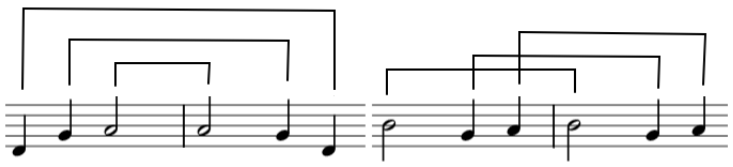
Typ-2 (Bezkontextové jazyky)

Tieto jazyky sa vyznačujú štruktúrami, ktoré sú reprezentované balancovanými stromami a ďalšími zrkadliacimi sa štruktúrami. Obrovské využitie týchto jazykov je u programovacích jazykoch, ktoré sú ich deterministický spracovateľnou podmnožinou. Jazyky typu-2 prinášajú so sebou nejednoznačnosť a neplatia u nich operácie doplnok, prienik a ďalšie. Tieto jazyky sú spracovateľné v $O(n^3)$ čase.

Určité časti hudobnej štruktúry, ktoré sú reprezentované hudobnými reťazcami vieme zaradiť medzi bezkontextové jazyky. Ide o motív alebo tému, ktorá postupne rastie a následne rovnakými tónami klesá. Tento jav tvorí zrkadlovú štruktúru, ktorá je zobrazená na ľavej strane obrázku 2.14.

Tieto gramatiky sú jednoduchšie na spracovanie oproti gramatikám, ktoré tvoria jazyky v nadchádzajúcich sekciách. Je to vďaka tomu, že umožňujú reťazce iba na jednej strane

produkčných pravidiel. Zložitosť je teda lineárna k počtu neterminálov v derivácií. Sila bezkotextových gramatik leží v tom, že umožňujú reprezentáciu viacúrovňových vnorených syntaktických štruktúr. Neterminal, ktorý môže reprezentovať motív, frázu, vetu alebo sekciu generuje reťazec tokenov na nižšej úrovni. Schopnosť generovať reťazce tokenov v jednom z produkčných pravidiel, ich robí lepším kandidátom k popisu hudobnej štruktúry, narozdiel od regulárnych gramatík.



Obr. 2.14: Znázornenie vnorených a kopírujúcich sa závislosti na variáciách.

V tejto kategórii sa nachádzajú jazyky tvorené gramatikami 2.2.3 a 2.2.6. Tieto gramatiky stačia pre popis len niektorých části hudobných pasáži, ktoré sa vyskytujú len v určitých častiach hudby. Jednou štruktúrovanou časťou sú variácie základnej témy retrogradáciou, ktorú môžeme vidieť na obrázku 2.10. Druhou časťou sú úseky hudby, ktoré nie sú štruktúrované do témy a jej variácie. Napriek tomu, že nie sú štruktúrované do témy a variácií tvoria reťazce, ktoré sú bezkontextové. Príkladom v hudbe môžu byť rôzne vsuvky, medzihry alebo vyhrávky. Tieto hudobné pasáže majú melodický a ozdobný charakter, ktoré sú charakterizovateľné vnorenými závislosťami bezkontextových jazykov.

Typ-1 (Kontextové jazyky)

Kontextové jazyky zahrňujú v sebe aj kopírovacie jazyky. Tieto jazyky sú rozhodnutelné, ale vyžadujú si vysokú výpočtovú silu vo väčšine prípadov. Ukázalo sa, že v niektorých prípadoch hudobné reťazce si vyžadujú silnejší formalizmus, ako sú bezkontextové gramatiky. Je to vďaka tomu, že tóny sú medzi jednotlivými taktami usporiadané vo forme kopírovacích jazykov. Ukážka takéhoto jazyka v notovom zápise je na pravej strane obrázka 2.14. Zápis tohto jazyka by mohol vyzerať ako $h^n g^n a^n$. Vo väčšine prípadov bude n nastavené na 2. Viac opakovaní nie je úplne zmysluplné, pokiaľ nechceme aby zneli tieto dva takty točili v slučke.

S kontextovými gramatikami prichádzajú aj problémy. Prvý problém je, že reťazce vygenerovane touto gramatikou sú nerozhodnutelné. Nerozhodné pravidlá sú súčasťou tejto gramatiky, čo znemožňuje zachovanie rovnakej frázovej štruktúry v reťazcoch generovaných kontextovou gramatikou. To je spôsobené tým, že každá strana z produkčných pravidiel môže byť reťazec tokenov. Reťazce tokenov na každej strane znemožňujú aplikáciu kontextových gramatik v analýze hudobnej štruktúry. Bezkontextové gramatiky môžu mať tiež nerozhodné pravidlá, ale krokovanie ich derivácií je zjednodušené. Zjednodušením je hociaký terminál, ktorý sa redukuje na jediný neterminal narozdiel od kontextových gramatik. U kontextových gramatík sa môže redukovať na celý reťazec.

Druhým problémom je implementácia. Použitie prekladača na takúto gramatiku spôsobuje znásobenie počtu produkčných pravidiel počtom kontextových možnosti. Špecifikácia takejto gramatiky nie je jednoducha. Popisovanie krokov gramatiky takého prekladača sa stane kombinatorické, pretože krížové závislosti musia byť vložené do produkčných tabuliek.

Napriek všetkým týmto komplikáciám je stále možné kontextové závislosti zabudovať do gramatiky konštruovanej čisto z bezkontextových produkčných pravidiel. Tento spôsob je možné celkom jednoznačne preložiť. Problém môže nastať u zobrazovaní kopírovacích závislostiach v gramatike.

Gramatika 2.2.2 generujúca jazyky, ktoré patria medzi kontextové sa ukázala ako vhodný kandidát pre popis hudobnej štruktúry. Vďaka tomu, že na oboch stranách pravidiel vieme dosadiť reťazce, potom sa dá jednoducho popísať notový zápis na pravej strane na obrázku 2.14. Deriváciami kontextovej gramatiky vieme pokryť tvorbu všetkých variácií. Nedeterministickosť kontextových gramatík neobmedzuje tvorby takej štruktúry, ako je téma a jej variácie. Práve naopak je táto vlastnosť vítaná, pretože nezväzuje ruky tvorcovi tejto štruktúry. Táto gramatika bude musieť umožniť tvorbu každej križiacej sa závislosti, ktoré obsahujú variácie. Konkrétna gramatika, ktorá bola použitá pre popis tejto štruktúry bude rozoberaná v nasledujúcej kapitole.

Typ-0 (Rekurzívne spočítatelné jazyky)

Rekurzívne spočítatelné jazyky sú jazyky, ktoré sú tvorené všeobecnými gramatikami. Tieto gramatiky nezavádzajú žiadne obmedzenia na produkčné pravidla. Z definície tieto gramatiky umožňujú vznik nekonečných reťazcov, ktoré samotné v hudbe nepredstavujú zmysel a nevedú tak k rozumnému riešeniu. Existujú ale hudobné prostredia, ktoré sa zaoberajú generovaním a syntézou v reálnom čase. Tieto prostredia sú turingovo úplne a volia si silnú generatívnu kapacitu. Problém s výpočtovou náročnosťou nechávajú v rukách užívateľa. Definícia 2.2.1 definuje gramatiku, ktorá generuje tieto jazyky. Využitie tejto gramatiky nie je momentálne v tejto práci, ale využitie môžu nájsť v jej ďalšom vývoji tejto práce.

Kapitola 3

Návrh modelu pre tvorbu nových hudobných pasáži

Obsahom tejto sekcie je návrh nových verzii modelov, ktoré by sa mohli aplikovať v hudbe. Návrh vychádza z gramatík s rozptýleným kontextom, ktoré sa ukázali vhodné v mnohých aspektoch tvorby hudby. Ako obrovská výhoda sa ukázala preskočenie kontextu, ktorý sa v rôznych pasážiach hudby vyskytuje. Ich ďalšou výhodou je, že umožňujú generovanie bezkontextových jazykov, ale i kontextových jazykov. Gramatiky s rozptýleným kontextom našli aplikácie v lingvistike. Aplikácia týchto gramatik v lingvistike je podobného princípu, ako sú aplikácie v hudbe. Na základe toho sa ukázali gramatiky s rozptýleným kontextom vhodné pre aplikáciu v hudbe. Teória ku gramatikam s rozptýleným kontextom a ich aplikáciam je prevzatá z knižiek [11, 12].

3.1 Gramatika s rozptýleným kontextom

Definícia 3.1.1 *Gramatikou s rozptýleným kontextom rozumieme štvoricu* $G = (V, T, P, S)$, *ktorá sa skladá z*

- *úplnej abecedy* V
- *množiny terminálov* T, *pre ktorú platí* $T \subset V$
- *konečnej množiny pravidiel* P *v tvare*

$$(A_1, ..., A_n) \to (x_1, ..., x_n),$$

  kde platí $n \geq 1, A_i \in V \setminus T, x_i \in V^*$ *pre všetky* i $1 \leq i \leq n$.
- *počiatočného symbolu* S *pre ktorý platí* $S \in V \setminus T$

V prípade, že

$$u = u_1 A_1 ... u_n A_n u_{n+1},$$

$$v = u_1 x_1 ... u_n x_n u_{n+1},$$

ako aj $p = (A_1, ..., A_n) \to (x_1, ..., x_n) \in P$, kde $u_i \in V^*$, pre každé i ohraničené podmienkou $1 \leq i \leq n+1$, máme derivačný krok od u po v na základe p vytvorený gramatikou G, ktorý zapisujeme $u \Rightarrow_G v\,[p]$, alebo zjednodušenou variantou $u \Rightarrow_G v$.

Ak platí $dlžka(p) \geq 2$, potom hovoríme, že p je kontextovým pravidlom. V prípade, že $dlžka(p) = 1$ tak p je bezkontextové pravidlo. Jazyk značený ako $L(G)$ je generovaný gramatikou G a definujeme ho ako

$$L(G) = \{x : x \in T^*, S \Rightarrow_G^* x\}.$$

Jazyk s rozptýleným kontextom je jazyk L práve vtedy, ak vieme nájsť takú gramatiku s rozptýleným kontextom G, kedy platí $L = L(G)$. Gramatiky s rozptýleným kontextom generujú jazyky s rozptýleným kontextom, ktoré označujeme ako JRC.

Príklad 3.1.1 *Zoberme si jazyk typu-1, ktorý je zobrazený na obrázku 2.13* $L = \{a^n b^n c^n | n \geq 2\}$. *Aby sme mohli tento jazyk generovať gramatikou v hudbe je potrebné ho zapísať pomocou tónov. Tento jazyk vyzerá nasledovne:* $L = \{c^n d^n e^n | n \geq 2\}$. *Aby sme vygenerovali tento jazyk pomocou gramatiky s rozptýleným kontextom definujeme ju nasledovne:*

$$G = (\{S, X, c, d, e\}, \{c, d, e\}, P, S)$$

kde

$$P = \{(S) \to (ccXddXeeX), (X, X, X) \to (cX, dX, eX), (X, X, X) \to (\epsilon, \epsilon, \epsilon)\}.$$

Majme užívateľa, ktorý by chcel vytvoriť hudobnú myšlienku v tvare cccdddeee. Myšlienka by bola vytvorená nasledovne:

$$S \Rightarrow_G ccXddXeeX \Rightarrow_G cccXdddXeeeX \Rightarrow_G cccdddeee.$$

Samotná gramatika s rozptýleným kontextom sa ukázala ako vhodná pre generovanie rôznych hudobných jazykov. Pod týmito jazykmi rozumieme JRC. Tento prístup je vhodný v prípade, že užívateľ vie ako by mal výstup vyzerať. No v prípade hudobníkov to vždy tak nie je. Predstavme si hudobníka, ktorý si nosí v hlave nejakú tému skladby. Dopredu nevie povedať ako bude vyzerať variácia tejto hudobnej myšlienky. Variácie existujú rôzne a vybrať si jednu, ktorá by sa mu hodila pre pokračovanie ďalej v skladbe vôbec nie je jednoduché. Zložitosť predstaviť si takú variáciu narastá s dĺžkou témy, od ktorej by sa mala rozvíjať. Princíp tohto riešenia môžeme spozorovať na pravej strane obrázku 2.14. Kedy by sa mohla v prvom takte nachádzať nejaká myšlienka skladateľa, a na druhej strane jej obmenenie, čiže variácia. Hovoríme teda o spomínanom kopírovacom jazyku, a krížiacich sa závislostiach. Krížiace sa závislosti je možné reprezentovať gramatikami s rozptýleným kontextom. Príklad gramatiky, ktorá vytvorí reťazec z obrázka je nasledovný.

Príklad 3.1.2 *Reťazec tónov z obrázka 2.14 sa skladá v dvoch taktoch z tónov hgahga. Majme gramatiku*

$$G = (\{S, X, h, g, a\}, \{h, g, a\}, P, S)$$

kde

$$P = \{(S) \to (XX), (X, X) \to (hX, hX)$$
$$(X, X) \to (aX, aX), (X, X) \to (gX, gX), (X, X) \to (\epsilon, \epsilon)\}.$$

Postup tvorby spomínaných dvoch taktov je nasledovný:

$$S \Rightarrow_G XX \Rightarrow_G hXhX \Rightarrow_G hgXhgX \Rightarrow_G hgaXhgaX \Rightarrow_G hgahga.$$

Na príklade 3.1.2 môžeme spozorovať, že gramatika generuje jazyk, ktorý nie je bezkontextový. Tento jazyk je JRC. Vytvorený reťazec demonštruje, že týmto spôsobom vieme vytvoriť požadovanú štruktúru téma a variácie.

Môžeme si všimnúť, že stále zanedbávame dĺžky jednotlivých tónov. Riešenie bude prezentované v nadchádzajúcej kapitole.

3.2 Spracovanie nôt

V tejto sekcii si rozoberieme akým spôsobom sú ďalej reprezentované noty. Reprezentácia nôt v programe nie je vôbec jednoduchá. Sú to grafické značky a zapisujú sa do notového zápisu, ktorý je tiež grafický. Toto znemožňuje ich jednoduchú reprezentáciu ako v programe, ale i vo vstupnom súbore. Keď sa pozrieme do notového zápisu nejakého taktu tak vidíme samotnú značku. Ako prvé podľa jej tvaru vieme odčítať jej dĺžku či ide o celú, polovú, štvrťovú alebo osminovú notu. Ak chceme zistiť tón tak ho odčítame z pozície noty v notovej osnove. Preto boli zavedené nasledovné značenia pre jednotlivé noty.

	c1	d1	e1	f1	g1	a1	h1
celá	c1_c	d1_c	e1_c	f1_c	g1_c	a1_c	h1_c
polová	c1_p	d1_p	d1_p	f1_p	g1_p	a1_p	h1_p
štvrťová	c1_š	d1_š	d1_š	f1_š	g1_š	a1_š	h1_š
osminová	c1_o	d1_o	d1_o	f1_o	g1_o	a1_o	h1_o

Tabuľka 3.1: Prehlad popisu nôt.

	c2	d2	e2	f2	g2	a2	h2	c3
celá	c2_c	d2_c	e2_c	f2_c	g2_c	a2_c	h2_c	c3_c
polová	c2_p	d2_p	e2_p	f2_p	g2_p	a2_p	h2_p	c3_p
štvrťová	c2_š	d2_š	e2_š	f2_š	g2_š	a2_š	h2_š	c3_š
osminová	c2_o	d2_o	e2_o	f2_o	g2_o	a2_o	h2_o	c3_o

Tabuľka 3.2: Prehlad popisu nôt.

Tabuľky 3.1 a 3.2 zobrazujú ako boli pomenované jednotlivé terminály, ktoré sú použité ďalej v práci. Stĺpce tabuľky sú pomenované po tónoch, ktoré sa nachádzajú na jednočiarkovanej a dvojčiarkovanej oktáve. Existuje množstvo tónov, ale tieto sú v bežných skladbách najpoužívanejšie. Riadky tabuľky sú pomenované po jednotlivých dĺžkach tónov, ktoré sú použité v tejto práci.

3.3 LL gramatika s rozptýleným kontextom

Táto sekcia sa zaoberá a definuje gramatiku s lineárnymi pravidlami. Ide o $LL(k)$ verziu gramatiky s rozptýleným kontextom. Pre získanie tejto gramatiky je potrebné modifikovať gramatiku s rozptýleným kontextom. Nižšie je rozoberaný spôsob v troch bodoch, ktorý popisuje tento proces. Obsahuje aj úpravy pre potrebu tvorby hudobných pasáži, ako sú variacie. Text, ale i definície v tejto kapitole sú prevzaté z [12].

1) Každé pravidlo gramatiky, ktoré by malo tvoriť hudobné pasáže sa musí skladať iba z lineárnych pravidiel. Pod tým rozumieme, že každé x_i môže obsahovať naraz iba jeden neterminál. Existuje ale jedna výnimka pre pravidlo pre počiatočný symbol S v tvare $(S) \rightarrow (x)$. Pod x rozumieme neprázdny reťazec, kde môže byť naraz k neterminalov. Kde k je pozitívne celé číslo pre každú gramatiku. Pre pravidlá ďalej platí, že S sa nesmie objaviť na pravej strane.

2) Majme pravidlo typu $(X, X) \rightarrow (c1\_c, c1\_c)$, ktoré vieme použiť pri kopírovaní tónov. Ak zoberieme prvé komponenty všetkých takýchto pravidiel vo forme $X \rightarrow c1\_c$,

potom získame bezkontextovu gramatiku, ktorá je LL gramatikou. Viac o LL gramatikách pozri [10].

3) Zoberme si variáciu opakovaním témy z obrázku 2.6. Aby sme prepísali terminály reprezentujúce tieto tóny $HGAHGA$, potrebujeme pravidlá aplikovať iba najľavejším spôsobom. Prepísanie pravidlom $(X, X, X) \to (h1\_c, g1\_š, a1\_š)$ je možné iba na $h1\_c\, g1\_š\, a1\_šHGA$. Teda $HHGGAA$ nie je možné prepísať týmto spôsobom. To nie je potrebné, pretože tieto neterminály nestvárňujú štruktúru téma a variácie.

Pred definovaním samotnej gramatiky je potrebné definovať dva pojmy.

Definícia 3.3.1 *Majme bezkontextovu gramatiku $G = (N, T, P, S)$. Potom pre každé $x \in (N \cup T)^*$ máme množinu*

$$prvý(G, x) = \{a \in T \mid x \Rightarrow_G^* ay\, vo\, G, y \in (N \cup T)^*\}.$$

Definícia 3.3.2 *Majme gramatiku s rozptýleným kontextom $G = (N, T, P, S)$. Značenie $pjadro(G)$ značí prvý komponent jadra gramatiky G. Definujeme ho ako bezkontextovu gramatiku*

$$pjadro(G) = (N, T, P', S),$$

kde

$$P' = \{A_1 \to x_1 \mid (A_1, A_2, ..., A_m) \to (x_1, x_2, ..., x_m) \in P'\}.$$

Definícia 3.3.3 *Majme gramatiku s rozptýleným kontextom $G = (N, T, P, S)$. Nech k je kladné celé číslo. Potom hovoríme o G ako o LL k-lineárnej gramatike s rozptýleným kontextom ak sú splnené tieto tri podmienky.*

1) Podmienka k-linearity. Pre každé pravidlo P platí, že je vo forme

$$(S) \to (x),$$

kde platí $x \in (N \setminus \{S\})^+, |x| \leq k$, alebo

$$(A_1, A_2, ..., A_m) \to (x_1, x_2, ..., x_m).$$

V pravidlách tohto typu platí, že $A_i \in (N \setminus \{S\}), x_i \in T^*(N \setminus \{S\})T^* \cup T^+$, pre všetký i ohraničené $1 \leq i \leq m$, pre nejaké $m \geq 1$.

2) Prvá LL podmienka. Majme relaciu $\Rightarrow_G$, ktorá je definovaná derivačným krokom od u po v, značená ako $u \Rightarrow_G v$. Vtedy a iba vtedy ak

$$u = u_1 A_1 u_2 A_2 ... u_m A_m y$$

$$v = u_1 x_1 u_2 x_2 ... u_m x_m y,$$

a

$$(A_1, A_2, ..., A_m) \to (x_1, x_2, ..., x_m) \in P.$$

kde $y \in V^*, u_i \in T^*$, pre všetky i, ktoré sú ohraničené $1 \leq i \leq m$.

3) Druhá LL podmienka. Majme dve rozdielne pravidlá

$$(A_1, A_2, ..., A_m) \to (x_1, x_2, ..., x_m) \in P$$

a

$$(B_1, B_2, ..., B_n) \to (y_1, y_2, ..., y_n),$$

ktoré spĺňajú podmienku $A_1 = B_1$, platí že

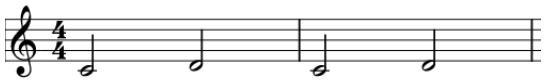
$$prvý(pjadro(G), x_1) \cap prvý(pjadro(G), y_1) = \emptyset$$

Príklad 3.3.1 *Majme LL 2-lineárnu gramatiku s rozptýleným kontextom* $G = (\{S, X\}, \{c1\_p, d1\_p, |\}, P, S)$*, kde*

$$P = \{(S) \to (XX), (X, X) \to (c1\_pX, c1\_pX),$$

$$(X, X) \to (d1\_pX, d1\_pX, (X, X) \to (|, |)\}.$$

potom tvorba nasledujúcej dvojice taktov z obrázku 3.1 vyzerá nasledovne:



Obr. 3.1: Variácia opakovaním.

$$S \Rightarrow_G XX \Rightarrow_G c1\_pXc1\_pX \Rightarrow_G c1\_pd1\_pXc1\_pd1\_pX \Rightarrow_G c1\_pd1\_p|c1\_pd1\_p|.$$

Zjavne platí, že $|XX| = 2$*, na pravej strane pravidiel na nevyskytuje S.*

Aby bolo možné správne reprezentovať tvorbu variácií pomocou gramatiky, budeme musieť vytvoriť novú gramatiku, ktorá bude obsahovať určité úpravy. Preto gramatika pre tvorbu variácií je nasledovná:

Definícia 3.3.4 $G = (\{S, X, R, c1\_c, d1\_c, e1\_c, f1\_c, g1\_c, a1\_c, h1\_c, c2\_c, d2\_c, e2\_c, f2\_c, g2\_c, a2\_c, h2\_c, c3\_c, c1\_p, d1\_p, d1\_p, f1\_p, g1\_p, a1\_p, h1\_p, c2\_p, d2\_p, e2\_p, f2\_p, g2\_p, a2\_p, h2\_p, c3\_p, c1\_š, d1\_š, d1\_š, f1\_š, g1\_š, a1\_š, h1\_š, c2\_š, d2\_š, e2\_š, f2\_š, g2\_š, a2\_š, h2\_š, c3\_š, c1\_o, d1\_o, d1\_o, f1\_o, g1\_o, a1\_o, h1\_o, c2\_o, d2\_o, e2\_o, f2\_o, g2\_o, a2\_o, h2\_o, c3\_o, |\}, \{c1\_c, d1\_c, e1\_c, f1\_c, g1\_c, a1\_c, h1\_c, c2\_c, d2\_c, e2\_c, f2\_c, g2\_c, a2\_c, h2\_c, c3\_c, c1\_p, d1\_p, d1\_p, f1\_p, g1\_p, a1\_p, h1\_p, c2\_p, d2\_p, e2\_p, f2\_p, g2\_p, a2\_p, h2\_p, c3\_p, c1\_š, d1\_š, d1\_š, f1\_š, g1\_š, a1\_š, h1\_š, c2\_š, d2\_š, e2\_š, f2\_š, g2\_š, a2\_š, h2\_š, c3\_š, c1\_o, d1\_o, d1\_o, f1\_o, g1\_o, a1\_o, h1\_o, c2\_o, d2\_o, e2\_o, f2\_o, g2\_o, a2\_o, h2\_o, c3\_o, |\}, P, S)$, *kde*

$$P = \{\{(S) \to (XX)\} \cup$$

$$\{(X, X) \to (c1\_cX, c1\_cX), (X, X) \to (d1\_cX, d1\_cX)$$

$$\vdots$$

$$(X, X) \to (c1\_pX, c1\_pX), (X, X) \to (d1\_pX, d1\_pX)$$

$$\vdots$$
$$(X,X) \to (c1\_\check{s}X, c1\_\check{s}X), (X,X) \to (d1\_\check{s}X, d1\_\check{s}X)$$
$$\vdots$$
$$\} \cup$$
$$\{(X,X) \to (c1\_cX, d1\_cX), (X,X) \to (d1\_cX, e1\_cX)$$
$$\vdots$$
$$(X,X) \to (c1\_pX, d1\_pX), (X,X) \to (d1\_pX, e1\_pX)$$
$$\vdots$$
$$(X,X) \to (c1\_\check{s}X, d1\_\check{s}X), (X,X) \to (d1\_\check{s}X, e1\_\check{s}X)$$
$$\vdots$$
$$\} \cup$$
$$\{(X,X) \to (d1\_cX, c1\_cX), (X,X) \to (e1\_cX, d1\_cX)$$
$$\vdots$$
$$(X,X) \to (g1\_oX, f1\_oX), (X,X) \to (a1\_oX, g1\_oX)$$
$$(X,X) \to (h1\_oX, a1\_oX), (X,X) \to (c2\_oX, h1\_oX)$$
$$\vdots$$
$$\} \cup$$
$$\{(X,X) \to (c1\_pX, c1\_cX), (X,X) \to (e1\_pX, e1\_cX)$$
$$\vdots$$
$$(X,X) \to (g1\_oX, g1\_\check{s}X), (X,X) \to (a1\_oX, a1\_\check{s}X)$$
$$(X,X) \to (h1\_oX, h1\_\check{s}X), (X,X) \to (c2\_oX, c2\_\check{s}X)$$
$$\vdots$$
$$\} \cup$$
$$\{(X,X) \to (c1\_cX, c1\_pX), (X,X) \to (e1\_cX, e1\_pX)$$
$$\vdots$$
$$(X,X) \to (g1\_\check{s}X, g1\_oX), (X,X) \to (a1\_\check{s}X, a1\_oX)$$
$$(X,X) \to (h1\_\check{s}X, h1\_oX), (X,X) \to (c2\_\check{s}X, c2\_oX)$$
$$\vdots$$
$$\} \cup$$
$$\{(X,X) \to (c1\_cX, c2\_pX), (X,X) \to (d1\_cX, d2\_cX)$$

$$\vdots$$
$$\}\cup$$
$$\{$$
$$\vdots$$
$$(X,X)\to(c2\_oX,d1\_oX),(X,X)\to(f1\_oX,a1\_oX)$$
$$(X,X)\to(c2\_cX,d1\_cX)$$
$$\vdots$$
$$(X,X)\to(|,|)\}\cup$$
$$\{(S)\to R$$
$$\vdots$$
$$R\to g1\_šRg1\_š, R\to a1\_šRa1\_š$$
$$R\to h1\_šRh1\_š, R\to c2\_šRc2\_š$$
$$\vdots$$
$$R\to|\}$$

U výslednej gramatiky platí $|XX| = 2$. Na žiadnych z pravidiel sa na pravej strane nevyskytuje počiatočný symbol S. U tejto gramatiky si môžeme všimnúť, že je porušená druhá LL podmienka z definície 3.3.3. Zoberme si pravidlá

$$(X,X)\to(c1\_cX,c1\_cX),(X,X)\to(c1\_cX,d1\_cX).$$

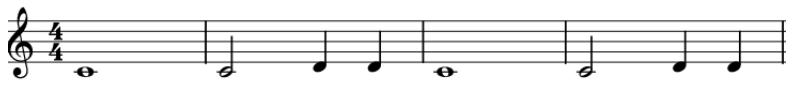
Prvé pravidlo patrí variácií kopírovaním a druhé pravidlo patrí variácií, ktorá postupne zvyšuje tému o jeden tón. Potom

$$prvý(pjadro(G),c1\_cX)\cap prvý(pjadro(G),c1\_cX)\neq\emptyset.$$

Porušenie tejto podmienky spôsobuje nejednoznačnosť pravidiel, z ktorých je konštruovaná výsledná LL tabuľka. Riešením tohto problému je, že jednotlivé skupiny pravidiel patria príslušným variáciám. Užívateľ, ktorý si zvolí tvorbu variácie opakovaním by zvolil prvé uvedené pravidlo. Zvolenie variácie pre postupne zvýšenie témy o jeden tón vyberie druhé pravidlo. Vďaka voľbe užívateľa nedochádza k takýmto konfliktom.

Definovaná gramatika obsahuje terminály, ktoré boli uvedené v tabuľkách 3.1 a 3.2. Kompletný zoznam pravidiel, ktoré boli použité v tejto práci je možné nájsť v prílohe. Vybrané pravidlá, ktoré sú uvedené vyššie boli použité na nasledujúce hudobné pasáže.

Príklad 3.3.2 *Majme variáciu opakovaním.*

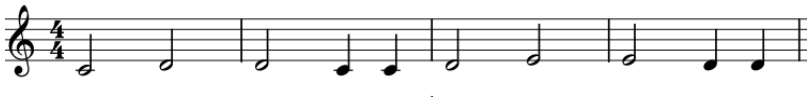


Obr. 3.2: Variácia opakovaním.

Aby sme mohli vytvoriť štvoricu taktov z obrázku 3.2 musí derivácia vyzerať nasledovne:

$$(S) \Rightarrow_G (XX)$$
$$\Rightarrow_G c1\_cXc1\_cX$$
$$\Rightarrow_G c1\_cc1\_pXc1\_cc1\_pX$$
$$\Rightarrow_G c1\_cc1\_pd1\_šXc1\_cc1\_pd1\_šX$$
$$\Rightarrow_G c1\_cc1\_pd1\_šd1\_šXc1\_cc1\_pd1\_šd1\_šX$$
$$\Rightarrow_G c1\_cc1\_pd1\_šd1\_š|c1\_cc1\_pd1\_šd1\_š|.$$

Príklad 3.3.3 *Majme variáciu, ktorá zvýši tému o jeden tón.*

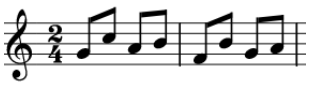


Obr. 3.3: Variácia zvýšením témy o jeden tón.

Vytvorenie tejto variácie zo štvorice taktov 3.3 je sprevádzané nasledujúcou deriváciou:

$$S \Rightarrow_G XX$$
$$\Rightarrow_G c1\_pXd1\_pX$$
$$\Rightarrow_G c1\_pd1\_pXd1\_pe1\_pX$$
$$\Rightarrow_G c1\_pd1\_pd1\_pXd1\_pe1\_pe1\_pX$$
$$\Rightarrow_G c1\_pd1\_pd1\_pd1\_šXd1\_pe1\_pe1\_pd1\_šX$$
$$\Rightarrow_G c1\_pd1\_pd1\_pc1\_šc1\_šXd1\_pe1\_pe1\_pd1\_šd1\_šX$$
$$\Rightarrow_G c1\_pd1\_pd1\_pc1\_šc1\_š|d1\_pe1\_pe1\_pd1\_šd1\_š|.$$

Príklad 3.3.4 *Majme variáciu, ktorá zníži tému o jeden tón.*

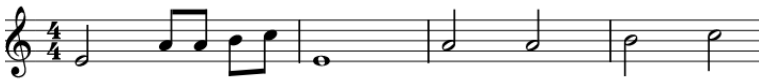


Obr. 3.4: Variácia znížením témy o jeden tón.

Pre tvorbu variácie z obrázku 3.4 je zostavená derivácia:

$$S \Rightarrow_G XX$$
$$\Rightarrow_G g1\_oXf1\_oX$$
$$\Rightarrow_G g1\_oc2\_oXf1\_oh1\_oX$$
$$\Rightarrow_G g1\_oc2\_oa1\_oXf1\_oh1\_og1\_oX$$
$$\Rightarrow_G g1\_oc2\_oa1\_oh1\_oXf1\_oh1\_og1\_oa1\_oX$$
$$\Rightarrow_G g1\_oc2\_oa1\_oh1\_o|f1\_oh1\_og1\_oa1\_o|.$$

Príklad 3.3.5 *Majme variáciu, ktorá predĺži tóny témy o polovicu.*

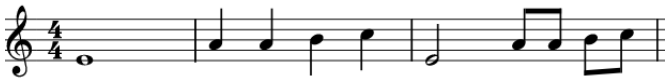


Obr. 3.5: Predĺženie dĺžky tónu o polovicu.

Derivácia variácie z obrázku 3.5 je nasledovná:

$$S \Rightarrow_G XX$$
$$\Rightarrow_G e1\_pXe1\_cX$$
$$\Rightarrow_G e1\_pa1\_oXe1\_ca1\_šX$$
$$\Rightarrow_G e1\_pa1\_oa1\_oXe1\_ca1\_ša1\_šX$$
$$\Rightarrow_G e1\_pa1\_oa1\_oh1\_oXe1\_ca1\_ša1\_šh1\_šX$$
$$\Rightarrow_G e1\_pa1\_oa1\_oh1\_oc2\_oXe1\_ca1\_ša1\_šh1\_šc2\_šX$$
$$\Rightarrow_G e1\_pa1\_oa1\_oh1\_oc2\_o|e1\_ca1\_ša1\_šh1\_šc2\_š|.$$

Príklad 3.3.6 *Majme variáciu, ktorá skráti dĺžku tónov témy.*

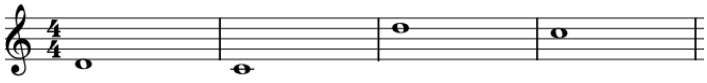


Obr. 3.6: Skrátenie dĺžky tónu o polovicu.

Derivácia pre variáciu z obrázku 3.6 vyzerá nasledovne:

$$S \Rightarrow_G XX$$
$$\Rightarrow_G e1\_cXe1\_pX$$
$$\Rightarrow_G e1\_ca1\_šXe1\_pa1\_oX$$
$$\Rightarrow_G e1\_ca1\_ša1\_šXe1\_pa1\_oa1\_oX$$
$$\Rightarrow_G e1\_ca1\_ša1\_šh1\_šXe1\_pa1\_oa1\_oh1\_oX$$
$$\Rightarrow_G e1\_ca1\_ša1\_šh1\_šc2\_šXe1\_pa1\_oa1\_oh1\_oc2\_oX$$
$$\Rightarrow_G e1\_ca1\_ša1\_šh1\_šc2\_š|e1\_pa1\_oa1\_oh1\_oc2\_o|.$$

Príklad 3.3.7 *Majme variáciu, ktorá vykoná transpozíciu témy.*



Obr. 3.7: Postupné zníženie témy o jeden tón.

Derivácia pre transpozíciu z obrázku 3.7 je nasledovná:

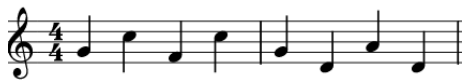
$$S \Rightarrow_G XX$$

$$\Rightarrow_G d1\_cXd2\_cX$$

$$\Rightarrow_G d1\_cc1\_cXd2\_cc2\_cX$$

$$\Rightarrow_G d1\_cc1\_c|d2\_cc2\_c|$$

Príklad 3.3.8 *Majme variáciu, ktorá vykoná protipohyb témy.*



Obr. 3.8: Variácia protipohybom.

Vzdialenosti pre variáciu protipohybom na obrázku sú $+3, -1, +3$. *Vo výslednej variácií sú vzdialenosti oproti prvému tónu* $-3, +1, -3$.

$$S \Rightarrow_G XX$$

$$\Rightarrow_G g1\_šXg1\_šX$$

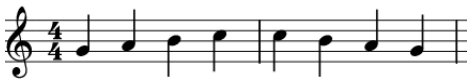
$$\Rightarrow_G g1\_šc2\_šXg1\_šd1\_šX$$

$$\Rightarrow_G g1\_šc2\_šf1\_šXg1\_šd1\_ša1\_šX$$

$$\Rightarrow_G g1\_šc2\_šf1\_šc2\_šXg1\_šd1\_ša1\_šd1\_šX$$

$$\Rightarrow_G g1\_šc2\_šf1\_šc2\_š|g1\_šd1\_ša1\_šd1\_š|$$

Príklad 3.3.9 *Majme variáciu, ktorá vykoná retrogradáciu témy.*



Obr. 3.9: Variácia retrogradáciou.

Dvojica taktov na obrázku 3.9 je vytvorená deriváciou:

$$S \Rightarrow_G R$$

$$\Rightarrow_G g1\_šRg1\_š$$

$$\Rightarrow_G g1\_ša1\_šRa1\_šg1\_š$$

$$\Rightarrow_G g1\_ša1\_šh1\_šRh1\_ša1\_šg1\_š$$

$$\Rightarrow_G g1\_ša1\_šh1\_šc2\_šRc2\_šh1\_ša1\_šg1\_š$$

$$\Rightarrow_G g1\_ša1\_šh1\_šc2\_š|c2\_šh1\_ša1\_šg1\_š$$

3.4 Nelimitovaný hlboký zásobníkový automat

Táto sekcia definuje nelimitovaný hlboký zásobníkový automat. Z jeho definície vychádzame pri návrhu tvorby algoritmu v nasledujúcej sekcii 3.5. Sekcia oboznamuje čitateľa s jeho využitím pri tvorbe variácií. Popisuje jeho novú verziu modelu, pomocou ktorého tvoríme variácie témy. Popisuje taktiež aj kľúčovú vlastnosť tohto automatu, ktorá je potrebná pre tvorbu variácií pomocou tohto automatu. Poznatky a definícia tohto automatu sú prevzaté z [8].

3.4.1 Definícia

Definícia 3.4.1 *Pod absolútne nelimitovaným zásobnikovým automatom rozumieme osmicu* $M = (Q, \Sigma, \Gamma, \$, R, s, S, F)$*, ktorá sa skladá z*

- *konečnej množiny stavov* Q*,*
- *vstupnej abecedy* Σ*,*
- *zásobnikovej abecedy* Γ*,*
- *množiny pravidiel* R*, ktoré sú konečnou reláciou* $R \subseteq Q \times (\Gamma \setminus (\Sigma \cup \{\$\})) \times Q \times (\Gamma \setminus \{\$\})^* \times Q \times (\Gamma \setminus \{\$\})^*)\{\$\})$*,*
- *špecialneho symbolu* $\$ \in \Gamma \setminus \Sigma$ *pre spodok zásobnika,*
- *počiatočného stavu* $s \in Q$*,*
- *počiatočného zásobnikového symbolu* $S \in \Gamma$*,*
- *množiny konečných stavov* $F \subseteq Q$*.*

Pravidlá nezapisujeme $(a, A, q, x) \in R$, ale ako $pA \to qx$. Majme reláciu ${}^{a}_{p}\vdash$ nad $Q \times \Sigma^* \times (\Gamma \setminus \{\$\})^*)\{\$\}$ takú, že $(p, aw, az) {}^{a}_{p}\vdash (p, w, z)$. Pre uvedené symboly platí $p \in Q, a \in \Sigma, w \in \Sigma^*$ a $z \in \Gamma^*$. Hovoríme potom o operácií zásobníka *pop* pre M, ktorá odstráni $a \in \Sigma$ z (p, aw, az) na (p, w, z). Pod $(p, q, uAv)^{a}_{\epsilon} \vdash (p, q, uAv)[pA \to qx]$ rozumieme, že M expanduje svoj zásobník s absolútne nelimitovanou expanziou z (p, w, uAv) po (q, w, uxv) na základe $pAv \to qx$.

Jazyk, ktorý M prijíma pomocou prázdneho zásobníka, $L(M)_\epsilon$ je definovaný ako

$$L(M)_\epsilon = \{w \in \Sigma^* \mid (s, w, S\$)^a \vdash^* (q, \epsilon, \$), kde\, q \in Q\}.$$

. Tento spôsob využívame aj v tejto práci.

3.4.2 Použitie a sila tvorby variácii

Hlboký zásobníkový automat definovaný pozri [12] je generalizáciou zásobníkového automatu 2.2.8. Využitie hlbokého zásobníkového automatu spočíva v LL syntaktickej analýze. V nej môžeme vrchný symbol zásobníka buď expandovať alebo ho odstrániť. Vďaka hlbokému zásobníkovému automatu môžeme túto expanziu vykonávať vo vnútri tohto zásobníka. Týmto sme zvýšili akceptačnú silu zásoníkového automatu. Vďaka tejto vlastnosti môžeme LL prekladačmi spracovať kontextové jazyky. Samozrejme, že toto neplatí pre všetky kontextové jazyky, pretože hĺbka expanzie je limitovaná.

Aby mohol užívateľ tvoriť variácie z nami navrhnutej gramatiky 3.3.4, budeme potrebovať hlboký zásobníkový automat. Dopredu však nepoznáme dĺžku reťazca, ktorý reprezentuje tému alebo celú skladbu. Aby sme mohli naďalej vytvárať variácie podľa našej

gramatiky, budeme potrebovať nami definovaný absolútne nelimitovaný hlboký zásobníkový automat. Ten nás nelimituje v hĺbke expanzií, takže môžeme vytvárať variácie o ľubovoľnej dĺžke.

Expanzie absolútne nelimitovaného zásobníkového automatu sú prirodzenou generalizáciou n–limitovaného hlbokého zásobníku. Pomocou pravidiel, ktoré majú špecifikovaný najvrchnejší neterminál vieme expandovať nezávisle na jeho hĺbke. Toto nám umožňuje aplikovať krížové vlastností variácií a vytvárať ich tak. Sila tvorby týchto variácií, ktoré vytvárame pomocou absolútne nelimitovaných hlbokých expanzií je rovnaká lineárne ohraničenému automatu pre propagujúcu sa verziu. Taktiež má aj silu turingovho stroja pre mazacie verzie.

3.4.3 Konštruovanie variácií

Podobný princíp, ktorý je použitý pre prekladač sme prevzali z [12]. Majme definovanú gramatiku pre tvorbu variácií 3.3.4. Aby sme mohli tvoriť variácie na základe reťazca, ktorý zadal užívateľ, musíme zaviesť absolútne nelimitovaný hlboký zásobníkový automat M. Ten zavedieme tak, že pre každé pravidlo s rozptýleným kontextom

$$(A_1, A_2, ..., A_n) \rightarrow (x_1, x_2, ..., x_n)$$

zavádzame pravidlá pre absolútne nelimitovaný zásobníkový automat

$$sA_n \rightarrow s_{n-1}x_n,$$

$$s_{n-1}A_{n-1} \rightarrow s_{n-2}x_{n-1},$$

$$\vdots$$

$$s_1A_1 \rightarrow sx_1.$$

M ma štartovný symbol s. Od s_1 po s_{n-1} máme nové unikátne stavy. M expanduje nevstupný symbol vo vnútri zásobníku, ktorým musí byť A. Expanzia A na x a presunutie sa do stavu q je podľa pravidla $pA \rightarrow qx$. Oproti hlbokému zásobníkovému automatu sme sa zbavili číslovania expanzií. Napriek tomu musíme dodržovať poradie expanzie, ktoré je uvedené vyššie. Ak by sme to nedodržali, tak by táto konštrukcia nefungovala správne.

	c1_c	|
S	1	1
X	2/3	4

Tabuľka 3.3: LL tabuľka

Zostrojená tabuľka 3.3 je ukážkou z gramatiky 3.3.4. Očíslované pravidlá sú

$$1 : (S) \rightarrow (XX)$$

$$2 : (X, X) \rightarrow (c1\_cX, c1\_cX), 3 : (X, X) \rightarrow (c1\_cX, d1\_cX)$$

$$4 : (X, X) \rightarrow (|, |).$$

Vo výslednej tabuľke uvažujeme iba o neterináloch, ktoré sú na ľavej strane. Nerozhodnosť, ktorá nastala medzi pravidlami 2 a 3 je rozhodnutá užívateľom. Ak si užívateľ zvolil variáciu opakovaním, zvolí sa pravidlo 2 inak 3.

Tento spôsob tvorby variácií podľa užívateľa a dodržania poradia expanzie zásobníka nám zanecháva hotové variácie. Zamerajme sa na pravú stranu pravidla 3. Tá sa skladá z dvoch tónov c1_c a d1_c. Tón c1_c reprezentuje tému a d1_c je tónom variácie. Tento princíp je aplikovaný u všetkých pravidiel. Takýmto spôsobom nahrádzame tóny témy na vrchole zásobníka a nelimitovane expandujeme zásobník M tónami variácie.

3.5 Algoritmus

Táto sekcia je venovaná návrhu algoritmu, vďaka ktorému budeme môcť pravidlami tvoriť výsledné variácie. Tento algoritmus je založený na poznatkoch z prác [6, 12]. Obe práce pracujú s iným spôsobom prekladania gramatík s rozptýleným kontextom. Ak hovoríme o využití oneskoreného vyhodnotenia, tak hovoríme o práci [6]. Ďalším spôsobom je použitie dočasného poľa a hlbokého zásobníkového automatu. Výsledný algoritmus sa zabera práve týmto spôsobom.

Tento spôsob nám umožňuje expanziu nie len na vrchole zásobníka, ale i hlboko v jeho vnútri. V tomto prípade je potrebné, aby zásobník bol implementovaný pomocou obojstranne viazaného zoznamu.

Algoritmus, ktorý nám umožňuje tvoriť variácie je založený na preklade bezkontextových gramatik pozri [10]. Počas tohto algoritmu pracujeme so zásobníkom ako obojstranne viazaným zoznamom. Každý symbol, ktorý vkladáme do zásobníka má tvar $\langle Symbol \rangle$. Vstup pre tento algoritmus je zostrojená LL-tabuľka na základe gramatiky 3.3.4. Ďalším vstupom je téma od užívateľa v podobe reťazca tokenov a požadované variácie.

Začiatok algoritmu (riadky 1–3) spočíva v inicializácií pomocných premenných. Jednou z nich je maximálny index poľa s variáciami, potom pozície pre pole variácií a vstupné tokeny. Poslednou pomocnou premennou je príznak pre expanziu vo vnútri zásobníka. Na riadku 4–5 máme inicializáciu samotného zásobníka a dočasného poľa.

Dočasné pole vieme limitovať dĺžkou 2. Je to vďaka tomu, že v jednom čase na základe našej gramatiky bude obsahovať maximálne odkazy na 2 neterminály. V poli každý element i ukazuje na jeden neterminál vo vnútri zásobníku. Na začiatku algoritmu je pole inicializovane tak, aby ukazovalo na prvý neterminál S.

Ďalej sa program vo vnútri `while` slučky vetvy na tri časti. Prvá na riadkoch 8–17 zahrňuje situáciu, kedy sa na vrchole zásobníka objaví znak pre ukončenie témy. Tento stav naznačuje to, že sa buď blížime ku koncu tvorby variácií alebo prechádzame na tvorbu ďalšej variácie. Stav prechodu na ďalšiu variáciu detekujeme tým, že sme nevyčerpali všetky variácie podľa pomocnej premennej. Výslednu variáciu uložíme a pokračujeme v tvorbe ďalšej variácie. Ak ide o ukončenie celej tvorby posunieme pozíciu premennej, ktorá číta tokeny zo vstupného reťazca. Po tomto stave sa nachádza na vstupe koniec témy a na zásobníku koncový symbol. Nasleduje tak uloženie poslednej variácie a tak vyprázdnime zásobník.

Druhá časť 18–20 je o čosi jednoduchšia. Tu detekujeme na vrchole zásobníka terminál, ktorý nie je koncom témy. Tento terminál odstránime z vrcholu zásobníka pomocou jeho operácie *pop*. Na vrchole zásobníka sa nahradí ďalší symbol a posunieme sa na ďalší vstupný token.

Tretia časť 23–37 nastáva vtedy, keď sa na vrchole zásobníka nachádza neterminál. Prvým krokom je získanie aktuálnej variácie. Potom sa začne s hľadaním pravidla v LL–tabuľke. Hľadanie prebieha na základe neterminálu X na vrchole zásobníka a terminálu t. Po získaní tohto pravidla je potrebné vybrať to, ktoré patrí k aktuálnej variácii. Ďalej mame podmienku, ktorá rozhoduje o expanzií zásobníka. Majme príznak pre expanziu zá-

sobníka v hĺbke je nastavený na *True*. Vyberieme z dočasného poľa pozíciu neterminálu z hĺbky zásobníka. Túto pozíciu odstránime z dočasného poľa. Neterminal na získanej pozícii nahradíme tónom variácie hlboko v zásobníku. Nastavíme príznak expanzie v hĺbke na *False*.

Ak je príznak nastavený na *False*. Získame pozíciu z dočasného poľa, ktorou je neterminál na vrchole zásobníka. Odstránime adresu tohto neterminálu z dočasného poľa. Zmeníme tento neterminal na vrcholu zásobníka za pretočenú časť pravidla pre variáciu. V prípade, že aktuálna variácia je retrogradácia, potom nepovolíme expanziu vo vnútri zásobníka.

V prípade variácie retrogradácia, je správanie sa tohto algoritmu podobné obyčajnému zásobníkovému automatu 2.2.8. Je to vďaka tomu, že túto variáciu vieme popísať bezkontextovou gramatikou. Počas tejto variácie sa nevykonáva expanzia vo vnútri zásobníku (Alg. 1 Riad. 36-37).

Koniec algoritmu nastáva vtedy, keď sme došli na koniec pola zadaných variácií. Funkcia *ulož_variáciu* nám postupne uložila tóny požadovaných variácií. Vstupný reťazec je prijatý, inak nastáva chyba. Reťazec je prijatý ako prvá časť jazyka, pod druhými časťami tohto jazyka rozumieme nami vytvorené variácie.

Algoritmus 1: Tabuľkou riadené spracovanie gramatiky s rozptýleným kontextom

Input: LL tabuľka pre $G = (N, T, P, S)$; $x \in T^*$; zvolené variácie
Output: variácie pre x, kde $x \in L(G)$ *a variácie* $\in L(G)$, inak *chyba*

```
1:  var_max_poz := len(variácie) − 1;
2:  var_poz, poz := 0;
3:  deep_push := False;
4:  inicializácia zásobníka s ⟨$⟩, ⟨S⟩;
5:  inicializácia dočastného poľa s odkazom na ⟨S⟩;
6:  while zásobník nie je prázdny:
7:      nech X je vrchol zásobníka, t je momentálny token;
8:      if X = |:
9:          var_poz += 1;
10:         if t = $ a zároveň var_poz > var_max_poz:
11:             ulož_variáciu();
12:             pop(X);
13:         elif t = | a zároveň var_poz > var_max_poz:
14:             position += 1;
15:         else:
16:             ulož_variáciu();
17:             reštart();
18:     elif X ∈ T:
19:         if t = X:
20:             pop(X);
21:             poz += 1;
22:         else:
23:             chyba;
24:     elif X ∈ N:
25:         var := získaj_variáciu(var_poz);
26:         r := {var : (X1, X2) → (x1, x2) ∈ LL–tabuľke[X, t]};
27:         odkaz na symbol z dočasného poľa vo vnútri zásobníka je P;
28:         if deep_push:
29:             vymeň v zasobniku P za otoč(r[x2]);
30:             dočas_pole.remove(P);
31:             deep_push = False;
32:         else:
33:             vymeň P za otoč(r[x1]);
34:             dočas_pole.remove(P);
35:             if variácia ≠ retrogradácia:
36:                 deep_push = True;
37:     else:
38:         chyba;
```

Kapitola 4

Implementácia

Kapitola implementácia sa zaoberá nami navrhnutým modelom pre tvorbu variácií. Implementácia tohto modelu vychádza z doteraz zozbieraných poznatkoch a vlastnostiach o bezkontextových, a kontextových gramatikách. Jadrom výslednej implementácie bude algoritmus 1. Pravidla, ktoré budú aplikované vychádzajú z gramatiky 3.3.4.

Najprv si rozoberieme technológie, ktoré sme si vybrali pre implementáciu. Zároveň uvedieme knižnicu, ktorá umožňuje prehrávanie výsledku. Potom zájdeme hlbšie, a to do štruktúry samotnej implementácie. Tá sa skladá z tried. Ďalej prezentujeme formát vstupu a výstupu, ktorý bol použitý pre reprezentáciu notového zápisu. Ako posledné si rozoberieme výsledky implementácie a jej porovnanie so zadanou témou.

4.1 Použité technologie a knižnice

Keďže predmetom tejto práce nie je kladený dôraz na rýchlosť prekladu, tak bol k implementácií použitý jazyk Python. Python je interpretovaný, objektovo orientovaný, dynamický typovaný a vysoko úrovňový programovací jazyk. Oproti jazykom akým je jazyk C alebo C++ postráda rýchlosť prekladu. Tá je kľúčová k implementácií prekladačov nových programovacích jazykov. Programovací jazyk python na oplátku ponúka jednoduchý syntax, obrovský výber a jednoduchú manipuláciu s balíkmi. Viac o tomto jazyku v [3].

Mimo štandardných balíčkov, implementovaný program používa jeden balíček, ktorý je pod všeobecnou verejnou licenciou. Ide o syntezátor, ktorého použitie si môže zvoliť užívateľ na prehrávanie výsledku tvorby variácií. Vďaka nemu si môže človek neznalý notového zápisu prehrať výsledne variácie. Ide o virtuálny analógový syntezátor. Návod na jeho manipuláciu a ukážky použitia môžeme nájsť v [13].

4.2 Vstup a výstup

Vstup aj výstup programu je vo formáte XML. Tento princíp je prevzatý z [16]. Tento formát nám umožňuje reprezentovať určitým spôsobom notový zápis. Tento formát sa skladá z elementov, ktorým vieme pripísať rôzne atribúty. To sa nám hodí pri popise notového zápisu, keďže jednotlivé tóny majú svoje atribúty. Notový zápis sa skladá taktiež z rôznych elementov (tónov) a atribútov (dĺžka). Príklad vstupného súboru je nasledujúci.

```
<theme time_signature="3/4">
    <measure>
        <tone note="half">c1</tone>
        <tone note="eight">d1</tone>
        <tone note="eight">e1</tone>
    </measure>
</theme>
```

Vstupný súbor obsahuje koreňový element *theme*. Tento element musí obsahovať atribút s predpisom dĺžky taktu *time_signature*. Aby sme predišli nezmyselným konštrukciám, tak sme tento atribút obmedzili na hodnoty 2/4, 3/4 a 4/4. Tieto predpisy taktov sú v hudbe najrozšírenejšie.

Vo vnútri koreňového elementu sú elementy *measure*. Ide o element pre takt. Ten obsahuje elementy pre jednotlivé tóny nazývane *tone*.

V notovom zápise je počet elementov *tone* závislí od taktového predpisu, ktorý je v tomto súbore 3/4. Tento element má atribút *note*, ktorý popisuje dĺžku tónu. Aby sme si overili dĺžku uvedeného elementu *measure*, potrebujeme spočítať dĺžky tónov v atribúte *note*. Maximálny počet dôb v takte je podľa *time_signature* 3. Zo štvorky vyčítame trvanie štvrťovej noty, ktoré je jednu dobu. Potom výpočet je nasledovný $2+1/2+1/2 = 3$. Polovicu doby trvá osminová nota, pretože štvrťová nota trvá jednu dobu. Textom elementov *tone* je názov tónu.

Na podobnom princípe je založený aj formát výstupu. Ten sa skladá z koreňového elementu *variations*. Vo vnútri tohto koreňového elementu sú zahrnuté všetky výsledné variácie. Tie sú zoradené podľa zoznamu argumentov uvedenom v podsekcii 4.3.1.

```
<?xml version="1.0" ?>
<variations>
        <variation>
                <measure>
                        <tone note="half">c1</tone>
                        <tone note="eighth">d1</tone>
                        <tone note="eighth">e1</tone>
                </measure>
        </variation>
</variations>
```

4.3 Prehľad tried

Výsledná implementácia využíva objektovo orientované programovanie. Štruktúra programu je rozdelená do tried. Tieto triedy vykonávajú rôzne činnosti, ako sú spracovanie argumentov, extrahovanie vstupu, tvorby variácií, prehrávanie variácií a vytvorenie výstupu.

4.3.1 Argumenty

Pomocou argumentov si užívateľ môže zvoliť variácie, ktoré sa budú vytvárať. Spracovanie argumentov zabezpečuje trieda *ArgParser*. Hlavnou metódou tejto triedy je *parse*. Cieľom tejto metódy je overiť správnosť argumentov a uložiť zadané argumenty pre ďalšie spracovanie. Jednotlivé variácie majú pridelené nasledujúce argumenty:

- -r, --repetition pre variáciu opakovanim
- -t, --transposition pre variáciu transpozíciou
- -S, --sequence-up pre postupne zvyšovanie o jeden tón
- -s, --sequence-down pre postupne zníženie o jeden tón
- -c, --contrary_motion pre variáciu proti pohybom
- -R, --retro_gradation pre variáciu retrogradáciou
- -a, --augmentation pre variáciu zdvojnásobenia dĺžky tónov
- -d, --diminution pre variáciu skrátenia dĺžky tónov o polovicu

Tieto argumenty patria všetkým variáciám, ktoré sme si navrhli a podarilo sa ich implementovať. Keďže sa pohybujeme na jednočiarkovanej a dvojčiarkovanej oktáve, nie všetky variácie sa môžu vykonávať viackrát pre jednu tému. Tie, ktoré vieme vykonať viackrát sú zvýšenie a zníženie témy o jeden tón. Patria pre ne obmedzenia, že sa môžu vykonať iba pre jednu oktávu. Takže pre tieto argumenty vieme zadať počet vykonaní od 1 po 7 vrátane. Ostatné argumenty iba povolia vykonanie príslušnej variácie. Argumenty

- -p, --play
- --save
- -v, --volume

súvisia s použitou knižnicou syntetizátora. Argument *-p* prehrá výsledné variácie, *--save* uloží výsledok variácií do súboru pre prehratie a *-v* nastaví hlasitosť výstupu. Argument -i a --input umožňuje zadať vstupný súbor vo formáte xml.

4.3.2 Spracovanie vstupu

Spracovanie vstupu zabezpečuje trieda *InputExtractor*. Jej úlohou je spracovať vstup zadaný užívateľom. Formát tohto vstupu je rozoberaný v 4.2.

Hlavnou metódou tejto triedy je *read_input*. Pomocou nej čítame vstup, z ktorého potrebujeme predpis taktov. Ten je potrebný pre formatovanie výstupu. Ďalej z elementov vo vstupnom súbore vyberie dĺžky tónov a ich názvy. Toto uskutočňuje pomocou metódy *fill_list*. Dvojice tón a dĺžka sú uložené do slovníka pre tvorbu variácií.

4.3.3 Zásobník

Aby sme mohli zásobník expandovať v nejakej hĺbke, potrebujeme ho implementovať pomocou obojstranne viazaného zoznamu. Informácie a časť metód pre obojstranne viazaný zoznam boli prevzaté z [4].

Implementácia zásobníka sa nachádza v súbore *DLL.py*. Súbor sa skladá z dvoch tried *Node* a *DLL*.

Trieda *Node* znázorňuje symbol v zásobníku. Tým môže byť hociktorý zo zásobníkovej abecedy. Súčasťou tohto symbolu sú odkazy na nasledujúci symbol a predchádzajúci symbol. Dôležitými operáciami tohto zásobníka sú:

def push(data)

Pomocou tejto operácie vkladáme prvé častí z pravých strán pravidiel LL gramatiky s rozptýleným kontextom. Tie sú vkladané na vrch zásobníka. Pod nimi rozumieme tóny hlavnej témy alebo skladby, ktorú chce užívateľ variovať.

def pop()

Vložené tóny na vrchu zásobníka sú rovnaké ako tie, ktoré si zadal užívateľ. Tie sú postupne z vrchu zásobníka odstraňované. Týmto spôsobom sa vyhneme tomu, že by výsledné variácie obsahovali tóny témy. Metóda nahradí na vrchu zásobníka nasledujúci symbol. Odkaz odstráneného symbolu je vráteny touto metódou.

def insert(prev_node, data)

Pomocou operácie *insert* máme možnosť expandovať zásobník i v jeho hĺbke. To nám umožňuje vložiť tóny variácie do vnútra zásobníka. Vkladané sú pravidlá variácie, ktoré sú súčasťou druhej časti pravej strany pravidiel LL gramatiky s rozptýleným kontextom. Tie sú ukladané za neterminál, ktorý je hĺbke zásobníka. Odkaz za tento neterminal je v parametri tejto metódy *prev_node*. Pomocou parametru *data* predáme symbol na vloženie.

def remove(node)

Aby sme mohli odstrániť neterminály v hĺbke zásobníka používame operáciu *remove*. Pre nahradenie neterminálu v hĺbke zásobníka musíme odstrániť po operácií *insert* neaktuálny neterminal. Adresa starého neterminálu je predaná parametrom *node* z dočasného poľa.

def printList(), def dll_len()

Trieda *DLL* obsahuje ďalšie pomocné metódy, ako sú *not_empty* pre zistenie či zásobník nie je prázdny. Obsahuje vypísanie obsahu *printList* a *dll_len* na získanie počtu symbolov v zásobníku.

4.3.4 Tvorba variácií

Tvorba variácií je sústredená do triedy *CSGparser*. Atribútmi tejto triedy sú tokeny podľa tabuľky 3.1. Ďalej kompletná LL tabuľka na základe gramatiky 3.3.4 a kompletný zoznam pravidiel. Tento kompletný zoznam pravidiel je uložený v poli *rules*.

def tokenizer()

Úlohou tejto metódy je tvoriť tokeny, ktoré sa skladajú z názvu tónu a jeho dĺžky. Ukladajú sa postupne do poľa *tokens*. Ich tvar je definovaný atribútmi triedy. Tieto tokeny sa tvoria zo slovníka, ktorý je vytvorený triedou *InputExtractor*. Na konci tohto procesu sa priložia tokeny pre taktovú čiaru a symbol, ktorý značí koniec tohto reťazca tokenov. Návratová hodnota je výsledný reťazec tokenov.

Metody pre prácu s pravidlami a tabuľkou

Aby sme mohli aplikovať pravidla podla algoritmu 1, museli sme zaviesť rôzne operácie nad LL tabuľkou.

Jednou z nich je *get_rule*. Táto metóda vyberie bunku z tabuľky na základe vstupného tokenu, ktorý zadal užívateľ a vrchného symbolu na zábobníku. Vo vnútri tejto bunky sú priradené pravidlá podľa variácie. Aby sme mohli vybrať správne pravidlo pre požadovanú variáciu máme metódy *get_key* a *get_rule_number*. Tieto metódy zvolia presné pravidlo pre variáciu z bunky tabuľky.

Pracujeme s obmedzenou množinou tónov, preto nemôžeme využiť plný potenciál pravidiel variácií. Ak nejaké pravidlo variácie presahuje túto množinu, tak sa zvoli pravidlo pre opakovanie tónu. Tieto vzťahy sú dopredu zavedené v LL tabuľke.

def reverse_push(items, deep, nonterminal)

Pomocou tejto metódy vieme pretočiť obsah pravej strany pravidiel LL gramatiky s rozptýleným kontextom. Tento otočený obsah pravidiel vkladáme do zásobníka. Vkladanie prebieha buď do hĺbky alebo na vrch zásobníka. Keďže pravidlá sa skladajú z n-tíc, ktoré sa nedajú metódami otočiť, tak sme použili radu z [18]. Pod parametrom *items* rozumieme obsah, ktorý sa vkladá do zásobníka. Odkaz na neterminál v zásobníku z dočasného poľa je *nonterminal* a *deep* rozhoduje o vkladaní do hĺbky.

def save_result(), def restart()

Obe metódy spolu úzko súvisia. Aby sme mohli uložiť jednotlivé úseky variácií, tak používame metódu *save_result*. Tá na konci zozbiera tóny vytvorenej variácie zo zásobníka a uloží ich. Aby sme mohli tvoriť ďalšie variácie, tak potrebujeme vynulovať pozície čítania vstupu od užívateľa. To robíme pomocou metódy *restart*. V prípade, že bol zadaný väčší počet variácií zvyšovania a znižovania tónov témy, potom sa variácie netvoria z témy. Tieto nasledujúce variácie sa tvoria z poslednej variácie zvyšovania a znižovania tónu.

def get_distances(tokens)

Aby sme mohli tvoriť variácie protipohybom, je potrebné získať vzdialenosti tónov zo vstupu. Tieto vzdialenosti sa počítajú na základe prvého tónu pre celú variáciu. Vzdialenosti sa vypočítajú vzorcom:

$$vzdialenosť = (prvý\_tón\_poz - tón\_poz) * 2.$$

Kde *prvý_tón_poz* je pozícia prvého tónu témy v poli *tuple_list* a *tón_poz* je pozícia každého ďalšieho tónu témy.

Príklad 4.3.1 *Majme tému z tónov f a e. Pozícia tónu f je 3 a pozícia tónu e je 2. Potom*

$$(3 - 2) * 2 = 2.$$

Výsledna variácia bude musieť posunúť tón e o 2 pozície hore. Preto výsledna variácia sa skladá z tónov f a g.

Návratová hodnota je zoznam týchto vzdialenosti. Tie sa použijú pri zadanej variácií protipohyb. Variácia protipohybom je tá, ktorá obsahuje najviac pravidiel gramatiky. Je to preto, že tu sa krížiace sa závislosti riadia vzdialenosťami tónov. Tie posúvajú tón hore o rovnakú dĺžku, ako ho posúvajú dole.

def variator()

Táto metóda je implementáciou algoritmu 1. Tvorí výsledné variácie, ktoré sa ukladajú do atribútu triedy *CSGparser*. Implementácia algoritmu je rozšírená o list vzdialenosti tónov pre variáciu protipohybom.

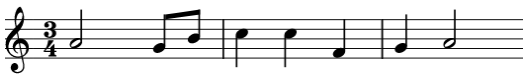
4.3.5 Prehranie výstupu a jeho výpis

Aby si užívateľ mohol prehrať vytvorené variácie, tak sme vytvorili triedu *TonesPlayer*. Cieľom tejto triedy je vytvoriť sínusové vlny podľa frekvencií, ktoré sme prevzali z [15]. Trvanie týchto vĺn je určené podľa dĺžky nôt, ktoré sú vo variácií. Aby sme mohli toto všetko prehrať a vytvoriť, použili sme knižnicu, ktorú sme rozoberali v sekcii 4.1. Tieto výsledky je možné prehrať, ale i uložiť do súboru s príponou *.wav*. Tento súbor sa uloží do rovnakého priečinku ako vstupný súbor.

O vypísanie výsledných variácii sa stará trieda *Generator*. Formát tohto výstupu sme rozoberali v sekcii 4.2. Výstupný xml formát je ešte formatovaný tak, aby nebol vypísaný na jednom riadku. O to sa stará knižnica *mini.dom*.

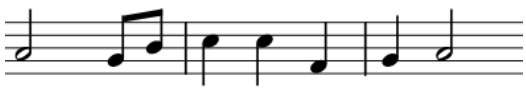
4.4 Výsledky a porovnanie

V tejto sekcií si predvedieme výsledky našej implementácie. Ukážeme si tému (vstup) a pre ňu vytvorené variácie (výstupy). Naša téma je jednoduchá a skladá sa z troch taktov. Zápis témy v notovom zápise s jej ekvivalentným zápisom v xml je nasledovný.



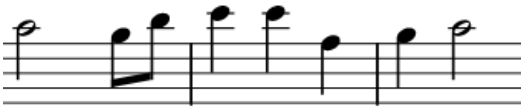
```
<theme time_signature="3/4">
    <measure>
        <tone note="half">a1</tone>
        <tone note="eight">g1</tone>
        <tone note="eight">h1</tone>
    </measure>
    <measure>
        <tone note="quarter">c2</tone>
        <tone note="quarter">c2</tone>
        <tone note="quarter">f1</tone>
    </measure>
    <measure>
        <tone note="quarter">g1</tone>
        <tone note="half">a1</tone>
    </measure>
</theme>
```

Ako prvú variáciu vytvoríme opakovanie témy. Táto variácia vznikla prekopírovaním tónov témy. Keď ju porovnáme s našou témou môžeme vidieť, že variácia sa vytvorila správne. Tóny variácie sú rovnaké ako tóny témy. Súčasťou tejto variácie sú krížiace sa závislosti so zadanou témou. Tie sme si znázornili na pravej strane obrázku 2.14. Spozorovať ich môžeme aj u ostatných variáciách až na retrogradáciu. Výsledok programu vo formáte xml a v notovom zápise je:



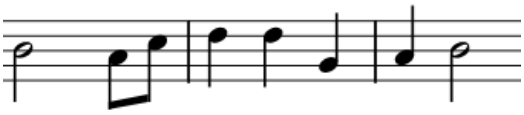
```
<variation>
    <measure>
        <tone note="half">a1</tone>
        <tone note="eighth">g1</tone>
        <tone note="eighth">h1</tone>
    </measure>
    <measure>
        <tone note="quarter">c2</tone>
        <tone note="quarter">c2</tone>
        <tone note="quarter">f1</tone>
    </measure>
    <measure>
        <tone note="quarter">g1</tone>
        <tone note="half">a1</tone>
    </measure>
</variation>.
```

Druhá variácia, ktorá bola vytvorená je transpozícia. Pomocou tejto variácie sme preniesli našu tému o oktávu vyššie. Ak porovnáme výsledok tejto variácie s témou, tak môžeme vidieť tóny z témy. Rozdiel je ten, že sú v dvojčarikovanej oktáve. Naša téma sa nachádza v jednočiarkovanej oktáve. Odčítať tento výsledok z notového zápisu môžeme z obrázku 2.1.



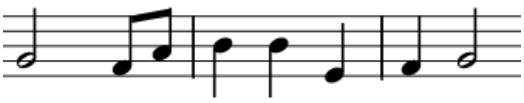
```
<variation>
    <measure>
        <tone note="half">a2</tone>
        <tone note="eighth">g2</tone>
        <tone note="eighth">h2</tone>
    </measure>
    <measure>
        <tone note="quarter">c3</tone>
        <tone note="quarter">c3</tone>
        <tone note="quarter">f2</tone>
    <measure>
        <tone note="quarter">g2</tone>
        <tone note="half">a2</tone>
    </measure>
</variation>
```

Nasledujúci výsledok je variácia postupným zvýšením tónu. Výsledné zvýšenie témy je o jeden tón. Túto variáciu sme vykonali jeden krát. Zoberme si tóny variácie a témy. Z 2.1 vieme vyčítať, že variácia prebehla úspešne. Overiť môžme výsledok aj sluchom. Vypočuť si tak môžeme stúpavú melódiu našej témy.



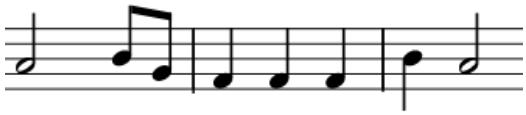
```
<variation>
    <measure>
        <tone note="half">h1</tone>
        <tone note="eighth">a1</tone>
        <tone note="eighth">c2</tone>
    </measure>
    <measure>
        <tone note="quarter">d2</tone>
        <tone note="quarter">d2</tone>
        <tone note="quarter">g1</tone>
    </measure>
    <measure>
        <tone note="quarter">a1</tone>
        <tone note="half">h1</tone>
    </measure>
</variation>
```

Ďalšia variácia je zníženie témy o jeden tón. Na základe porovnania témy a výsledku vieme z taktového zápisu, že sme znížili tému o jeden tón. Tento výsledok vieme overiť aj sluchovo. Tak zistíme, že naša téma postupne klesá. Ak by sme nepracovali ohraničené na jednočiarkovanej a dvojčiarkovanej oktáve, tak by táto variácia pre nejaké n mohla klesať ešte nižšie. To by platilo pre frekvencie, ktoré vieme zachytiť sluchom.



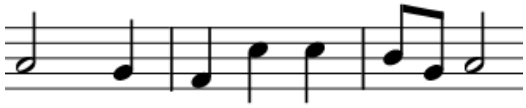
```
<variation>
    <measure>
        <tone note="half">g1</tone>
        <tone note="eighth">f1</tone>
        <tone note="eighth">a1</tone>
    </measure>
    <measure>
        <tone note="quarter">h1</tone>
        <tone note="quarter">h1</tone>
        <tone note="quarter">e1</tone>
    </measure>
    <measure>
        <tone note="quarter">f1</tone>
        <tone note="half">g1</tone>
    </measure>
</variation>
```

Nasleduje výsledok pre variáciu protipohybom. Aby sme ju mohli vytvoriť, uvažovali sme vzdialenosti tónov z témy -1, +1, +2, +2, -2, -1 a 0. Výsledné vzdialenosti sú +1, -1, -2, -2, -2, +1 a 0. Môžeme si všimnúť piatu pozíciu, ktorá sa nezmenila. Je to tak, pretože náš systém posúva tóny dole o hodnotu, ktorú môže posunúť tóny aj hore. Keďže toto nemôžeme vykonať, tak tento tón nemeníme. Uplatnilo sa tak pravidlo opakovania.



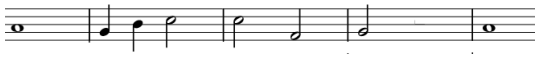
```
<variation>
    <measure>
        <tone note="half">a1</tone>
        <tone note="eighth">h1</tone>
        <tone note="eighth">g1</tone>
    </measure>
    <measure>
        <tone note="quarter">f1</tone>
        <tone note="quarter">f1</tone>
        <tone note="quarter">f1</tone>
    </measure>
    <measure>
        <tone note="quarter">h1</tone>
        <tone note="half">a1</tone>
    </measure>
</variation>
```

Preklopenie témy, ktoré sme vytvorili variáciou témy je ďalším výsledkom. Ak ju porovnáme s témou vidíme, že sa začína od jej konca. Takto sme vytvorili vnorené závislosti, ktoré sú typické pre bezkontextové jazyky. Ich znázornenie môžeme nájsť na ľavej strane obrázku 2.14. Výsledok je na nasledujúcom obrázku.



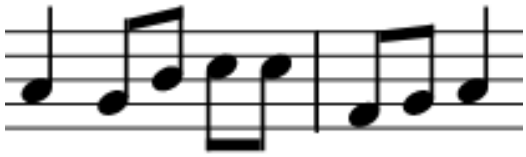
```
<variation>
    <measure>
        <tone note="half">a1</tone>
        <tone note="quarter">g1</tone>
    </measure>
    <measure>
        <tone note="quarter">f1</tone>
        <tone note="quarter">c2</tone>
        <tone note="quarter">c2</tone>
    </measure>
    <measure>
        <tone note="eighth">h1</tone>
        <tone note="eighth">g1</tone>
        <tone note="half">a1</tone>
    </measure>
</variation>
```

Posledne dve variácie, ktoré vieme vytvoriť manipulujú s dĺžkami nôt našej témy. Nasledujúca variácia úspešne predĺžila dĺžky nôt témy. Samotné tóny témy sa zachovali. Počet taktov sa rozšíril, keďže dĺžky tónov by sme do pôvodného počtu taktov nezmestili.



```
<variation>
    <measure>
        <tone note="full">a1</tone>
    </measure>
    <measure>
        <tone note="quarter">g1</tone>
        <tone note="quarter">h1</tone>
        <tone note="half">c2</tone>
    </measure>
    <measure>
        <tone note="half">c2</tone>
        <tone note="half">f1</tone>
    </measure>
    <measure>
        <tone note="half">g1</tone>
        <tone note="full">a1</tone>
    </measure>
</variation>
```

Opakom predchádzajúcej variácie je skrátenie dĺžky tónov témy. U tejto variácie sa dĺžka nôt skrátila natoľko, že sme ich mohli vložiť iba do dvoch taktov. V porovnaní s témou sme tóny zachovali, zmenili sme však dĺžky nôt. Dĺžku nôt témy sme skrátili na polovicu.



```
<variation>
    <measure>
        <tone note="quarter">a1</tone>
        <tone note="eighth">g1</tone>
        <tone note="eighth">h1</tone>
        <tone note="eighth">c2</tone>
        <tone note="eighth">c2</tone>
    </measure>
    <measure>
        <tone note="eighth">f1</tone>
        <tone note="eighth">g1</tone>
        <tone note="quarter">a1</tone>
    </measure>
</variation>
```

Kapitola 5

Záver

5.1 Dosiahnuté výsledky

Predmetom práce bolo zoznámiť sa s rôznymi formálnymi modelmi. Jedným z nich, ktoré našli uplatnenie v hudbe sú Markovovské procesy. Uplatnenie týchto modelov môžeme nájsť v práci [16]. Taktiež by sme vedeli nájsť rôzne uplatnenia pre automaty a gramatiky, ktoré sú rozoberané v tejto práci. Z pozorovania vyplynulo, že rôzne vlastností gramatík a automatov umožňujú popísať, a postupne pravidlami tvoriť rôzne druhy hudobných konštrukcií.

Štúdiom hudobných konštrukcii a úsekov sme zistili, že tie, ktoré by sme vedeli popísať formálnymi modelmi sú téma a jej variácie. Variácie, ktoré boli použité pre výsledné modely pochádzajú z sématického rozpoznávania hudobných objektov. Tieto variácie sú opakovanie, transpozícia, postupnosť, protipohyb, retrogradácia, predĺženie a skrátenie dĺžky tónu. Študovali sme tieto variácie. Hľadaním spôsobu, ako popísať tieto konštrukcie formálnymi modelmi sme zistili, že vhodný formálny model sú gramatiky. Rôzne variácie, ktoré sú zapísané v notovom zápise sa ukázali ako reťazce. Tieto reťazce vykazujú vlastnosti, ktoré vieme popísať bezkontextovými a kontextovými jazykmi. Hovoríme o vnorených závislostiach pre bezkontextové jazyky a krížiace sa závislosti pre kontextové jazyky. Variácia retrogradácia ukázala tvorbu hudobných reťazcov vnorenými závislosťami medzi tónmi. Výsledkom tohto pozorovania je bezkontextová gramatika, ktorá pravidlami popisuje túto konštrukciu. Vďaka tejto gramatike sme pravidlami vytvorili variáciu retrogradácia. Ostatné variácie sa ukázali ako nepopísateľné bezkotextovými gramatikami. Aby sme mohli ostatné variácie tvoriť museli sme zájsť do kontextových gramatik.

Kontextové gramatiky sú silnejší formalizmus akým sú bezkontextové gramatiky. Vďaka tomu sme mohli do nej zahrnúť aj tvorbu retrogradácie, ktorá je bezkotextova. Štúdium rôznych kontextových gramatík, ktoré by mohli byť vhodné pre tvorbu hudobnej štruktúry ukázalo, že vhodnou gramatikou bude gramatika s rozptýleným kontextom. Existujúce aplikácie gramatiky s rozptýleným kontextom v lingvistike ukázali možnosť generovania gramatických viet. Tento princíp sme uplatnili v hudbe. Preskočenie kontextu v hudbe nám umožnilo generovať rôzne reťazce. Pod týmito reťazcami chápeme variácie témy. Tieto variácie vytvárame programovo vstupom užívateľa. Zostavili sme teda prekladač na základe LL gramatiky s rozptýleným kontextom a absolútne nelimitovaným hlbokým zásobníkovým automatom. Ten nám umožní spracovať vstupný reťazec, ktorý je téma. Počas tohto procesu vytvorí variáciu k tejto téme. Tento proces je potrebný, pretože spolu tvoria jednu štruktúru. Výsledná LL tabuľka sa skladá z pravidiel, ktoré znázorňujú tvorbu jednotlivých križiacich sa závislosti vo variáciách. Výsledná tabuľka by bola nerozhodnuteľná, ale vďaka

tomu, že rozlišujeme pravidlá pre jednotlivé variácie, tak vieme zvoliť výsledné pravidlo pre konkrétnu variáciu.

Na základe týchto výsledkov sme si mohli zaviesť novú verziu modelu, ktorý je založený na LL gramatike s rozptýleným kontextom. Výsledkom tohto modelu je tvorba zvolenej variácie na základe vstupnej témy od užívateľa. Tento model sme implementovali v jazyku python. Pre správny chod tejto aplikácie je potrebný vstupný súbor vo formáte xml. Xml formát je použitý na reprezentáciu nôt pomocou elementov, a ich atribútov. Tento formát je vhodný aj pre výstup, ktorým sú pravidlami vyprodukované variácie. Jadro výslednej implementácie sa skladá z LL tabuľky, ktorá obsahuje pravidlá LL gramatiky s rozptýleným kontextom. Skladanie variácie je riadené nekonečným hlbokým zásobníkom, vďaka ktorému môžeme neustále rozširovať zásobník v hĺbke. Výsledné použitie tohto zásobníku leží v tom, že aplikuje prvú časť pravej strany pravidla ako pravidlo témy, a hlboko do zásobníka pravidlo variácie tónu z témy.

Dosiahnutý výsledok tejto práce spočíva vo variovaní rôznych tém, ktoré zadá užívateľ. Na základe vytvorených výstupov si vie pozrieť zápis, ale aj vypočuť výsledné variácie. To mu zjednoduší rozhodovanie o tom, ktorú variáciu použije a či ju použije vo výslednej skladbe. Zjednodušenie je aj v tom, že užívateľ nemusí odchádzať od počítača k overeniu si svojej predstavy na hudobnom nástroji. Nemusí si taktiež ani zapisovať variáciu aby si ju následne overil. Výsledok je zapísaný vo formáte xml. Ten sa jednoducho spracováva rôznymi knižnicami. Výsledok sa dá použiť k obmieňaniu hudby v počítačových hrách. Hudba v týchto hrách sa skladá z rôznych melódií. Táto práca ich vie variovať a priniesť tak rozmanitejší zážitok z hudby v hrách.

5.2 Ďalší vývoj

Ďalší vývoj tohto projektu sa môže uberať viacerými smermi. Jedným z nich môže byť zbieraním podkladov a analýza existujúcich variácií. To by mohlo viesť ešte ku komplexnejšiemu systému. Tento komplexnejší systém by sa mohol ďalej zaoberať absolútnym neohraničeným zásobníkom a jeho vlastnosťami. Výsledok by mohol priniesť nový model, ktorý tvorí hudbu v reálnom čase zo zozbieraných podkladov a vstupu užívateľa. Tvorená hudba by mohla znieť počas expanzie zásobníka.

Ďalším spôsobom môže byť tvorenie hudby v reálnom čase pomocou jazykov typu-0. Tento formálny model by sa už nemal odrážať od určitých úsekov, alebo objektov hudby ako tento projekt. Jeho výsledkom by mal byť popis hudby, ktorá je vhodná do nejakej situácie. Tá by sa mohla v reálnom čase meniť na základe vstupu užívateľa. Ten by si určoval vlastnosti hudby. Zaujímavá myšlienka môže byť pridanie pravdepodobností do tohto systému. Vďaka tomu vytvoriť rôzne melodické a uchu lahodiace konštrukcie v reálnom čase. Využitie tohto prístupu by existovalo aj napríklad v počítačových hrách.

Aplikácie v hudbe našli aj rôzne ďalšie formálne modely. Ďalšia práca by sa mohla zaoberať štúdiom rôznych formálnych modelov. Jedným z nich môžu byť pravdepodobnostné bezkontextové gramatiky. Vybraním sa týmto smerom prináša štúdium článku [5]. Ten sa zaoberá aplikovaním rôznych existujúcich modelov pre hľadanie štruktúry v hudbe. Tieto modely pochádzajú z aplikácií na bežný jazyk. Taktiež zaujímavý pohľad prináša článok [19]. Ten sa zaoberá štruktúrami, ktoré sa nachádzajú v jazyku, hudbe a určitých častiach zvieracích zvukoch. K týmto štruktúram sa snaží priradiť formálne modely pomocou Chomského hierarchie.

Literatúra

[1] Droppová, A. *Elementárna hudobná teória*. 1. vyd. Bural, Jovsa, 1998. 104 s. ISBN 80-88697-39-5.

[2] Dunnett, B. *Theme and Variations* [online]. 2013 [cit. 2022-03-07]. Dostupné z: https://www.musictheoryacademy.com/understanding-music/theme-and-variations/.

[3] Foundation, P. S. *What is Python? Executive Summary* [online]. [cit. 2022-04-28]. Dostupné z: https://www.python.org/doc/essays/blurb/.

[4] GeeksforGeeks. *Doubly Linked List | Set 1 (Introduction and Insertion)* [online]. Január 2022 [cit. 2022-04-29]. Dostupné z: https://www.geeksforgeeks.org/doubly-linked-list/.

[5] Gilbert Édouard a Conklin, D. A Probabilistic Context-Free Grammar for Melodic Reduction. *Proceedings of the International Workshop on Artificial Intelligence and Music, 20th International Joint Conference on Artificial Intelligence (IJCAI)*. Hyderabad, India: [b.n.]. 2007, s. 83–94.

[6] Jirák, O. *Table-driven parsing of scattered context grammar* [online]. Január 2010 [cit. 2022-04-25]. Dostupné z: https://www.researchgate.net/publication/228952574_Table-Driven_Parsing_of_Scattered_Context_Grammar.

[7] Jurish, B. *Music as a Formal Language* [online]. September 2004 [cit. 2022-03-26]. Dostupné z: https://www.researchgate.net/publication/228806126_Music_as_a_formal_language.

[8] Kučera, J., Meduna, A. a Soukup, O. Absolutely Unlimited Deep Pushdown Automata. In: *Proceedings of the 10th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2015)*. Ing. Vladislav Pokorný - Litera, 2015, s. 36–44. ISBN 978-80-214-5254-1. Dostupné z: https://www.fit.vut.cz/research/publication/10978.

[9] Meduna, A. *Automata and Languages: Theory and Applications [Springer, 2000]*. Springer Verlag, 2005. 892 s. ISBN 1-85233-074-0. Dostupné z: https://www.fit.vut.cz/research/publication/6177.

[10] Meduna, A. *Formal Languages and Computation*. Taylor & Francis Informa plc, 2014. 315 s. Taylor and Francis. ISBN 978-1-4665-1345-7. Dostupné z: https://www.fit.vut.cz/research/publication/10524.

[11] Meduna, A. a Techet, J. *Scattered Context Grammars and their Applications*. WIT Press, 2010. 224 s. WIT Press, UK. ISBN 978-1-84564-426-0. Dostupné z: https://www.fit.vut.cz/research/publication/8997.

[12] Meduna, A. a Zemek, P. *Regulated Grammars and Automata*. Springer US, 2014. 694 s. ISBN 978-1-4939-0368-9. Dostupné z: https://www.fit.vut.cz/research/publication/10498.

[13] Mihira, Y. *Python synthesizer* [online]. Február 2020 [cit. 2022-04-28]. Dostupné z: https://github.com/yuma-m/synthesizer.

[14] Roads, C. a Wieneke, P. *Grammars as Representations for Music* [online]. 1979 [cit. 2022-03-27]. Dostupné z: https://www.jstor.org/stable/3679756?read-now=1&refreqid=excelsior%3A5c22caafd6db13b89748ac8621d1818a&seq=3.

[15] Schnupp, J., Nelken, E. a King, A. *Fundamental frequencies of Notes in Western Music* [online]. [cit. 2022-03-21]. Dostupné z: https://auditoryneuroscience.com/pitch/fundamental-frequencies-notes-western-music.

[16] Schulze, W. *A Formal Language Theory Approach To Music Generation*. Matieland 7602, South Africa, 2009. Diplomová práca. University of Stellenbosch. Vedúci práce Merwe, A. van der.

[17] Shan, M.-K., Chiang, M.-F. a Kuo, F.-F. *Relevance feedback for category search in music retrieval based on semantic concept learning* [online]. 2008 [cit. 2022-03-20]. Dostupné z: https://www.researchgate.net/publication/220664738_Relevance_feedback_for_category_search_in_music_retrieval-_based_on_semantic_concept_learning.

[18] Stackoverflow.. *Traverse a list in reverse order in Python* [online]. Február 2009 [cit. 2022-04-29]. Otázka od: Joan Venge, Odpovedal: Greg Hewgill. Dostupné z: https://stackoverflow.com/questions/529424/traverse-a-list-in-reverse-order-in-python.

[19] Zuidema, W., Hupkes, D., Wiggins, G., Scharff, C. a Rohrmeier, M. Formal models of Structure Building in Music, Language and Animal Song. [online]. Január 2019, [cit. 2022-04-27]. Dostupné z: https://www.researchgate.net/publication/330439467_Formal_models_of_Structure_Building_in_Music_Language_and_Animal_Songs.

Príloha A

Obsah priloženého pamäťového média

```
/
├──rsc/                 - Vstupne a výstupné súbory.
│   ├── example.xml     - Ukážka vstupného súboru.
│   └── example.wav     - Zvuková ukážka vstupného súboru.
├──src/                 - Zdrojové súbory programu
│   ├── csgparser/      - Moduly.
│   └── variator.py     - Hlavný zdrojový súbor.
├──thesis/              - Text práce a zdrojové súbory k textu.
├──README.md            - Návod na inštaláciu a spustenie.
├──requirements.txt     - Potrebné baličky mimo štandardných balíčkov.
└──rules.txt            - Kompletný zoznam použitých pravidiel.
```