VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

TEORIE FORMÁLNÍCH JAZYKŮ APLIKOVANÁ V MUZIKOLOGII

FORMAL LANGUAGE THEORY APPLIED IN MUSICOLOGY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TEREZA STRAKOVÁ**
AUTHOR

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

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

141148

Ústav: Ústav informačních systémů (UIFS)
Studentka: **Straková Tereza**
Program: Informační technologie
Specializace: Informační technologie

Název: **Teorie formálních jazyků aplikovaná v muzikologii**
Kategorie: Teoretická informatika
Akademický rok: 2022/23

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

Při obhajobě semestrální části projektu je požadováno:
1, 2, 3.

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

Vedoucí práce: **Meduna Alexandr, prof. RNDr., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 31.7.2023
Datum schválení: 29.10.2022

Abstrakt

Cieľom tejto práce je aplikovať formálne modely v hudbe, konkrétne na generáciu hudobných reťazcov.

V práci bola navrhnutá gramatika s rozptýleným kontextom pre generáciu variácie hudobného motívu. Ďalej bol navrhnutý algoritmus pre tabuľkou riadené spracovanie týchto gramatík. Algoritmus využíva zoznam odložených pravidiel, čo umožňuje pracovať iba s vrcholom zásobníka. Algoritmus bol implementovaný ako aplikácia, ktorá umožňuje vygenerované motívy prehrať alebo uložiť.

Abstract

The aim of this thesis is to apply formal models in music. Specifically, it concentrates on the generation of musical strings.

In the thesis, a scattered context grammar for generating variations of a musical motif was designed. Next, an algorithm was designed for table driven parsing of the grammars. The algorithm uses a list of stored rules, which allows it to work with only the top of a stack. The algorithm was implemented as an application, which allows the generated motifs to be played or saved.

Kľúčové slová

formálne jazyky, gramatika s rozptýleným kontextom, hudba, téma a variácie, music21

Keywords

formal languages, scattered context grammar, music, theme and variations, music21

Citácia

STRAKOVÁ, Tereza. *Teorie formálních jazyků aplikovaná v muzikologii.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. RNDr. Alexander Meduna, CSc.

Teorie formálních jazyků aplikovaná v muzikologii

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením prof. RNDr. Alexandra Medunu CSc. Uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Tereza Straková
31. júla 2023

Poďakovanie

Ďakujem profesorovi Medunovi za jeho odborné vedenie a cenné rady pri vypracovaní tejto práce.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Formálne jazyky a gramatiky boli zavedené ako nástroje pre skúmanie prirodzeného jazyka. Uplatnenie našli v informatike, pri tvorbe programovacích jazykov a prekladačov. S postupom času formálne modely našli uplatnenie aj v iných odvetviach ako napríklad biológia alebo v rôznych druhoch umenia.

Cieľom tejto práce je ukázať aplikácie formálnych modelov v hudobnom umení, konkrétne pre generáciu variácií podľa zadaného motívu. V časti hudobnej teórie budú ukázané niektoré druhy variačných techník, ktoré je možné aplikovať na krátky motív alebo rozsiahlejšiu tému. Výsledná aplikácia bude implementovať generáciu týchto variácií.

Kapitola 4 predstavuje základy hudobnej teórie. Predstavuje najmä, čo tvorí melódiu a ako sa melódie zapisujú. Potom predstaví prácu s motívom a hudobné formy so zameraním na variačnú hudobnú formu.

V kapitole 3 sú najprv prezentované základné prostriedky pre definovanie formálnych jazykov. Potom kapitola uvádza formálne modely podľa Chomského hierarchie.

Kapitola 4 uvádza definíciu gramatík s rozptýleným kontextom, na ktoré sa táto práca zameriava. Potom sú v kapitole definované nové modely, ktoré budú použité pre generáciu hudobných reťazcov.

V kapitole 5 je popísaná štruktúra výslednej aplikácie, zvolené prostriedky pre implementáciu a implementácia samotná. Kapitola popisuje aj vlastný formát pre vstupné súbory. Nakoniec uvádza ukážky výstupov.

Kapitola 2

Základné pojmy hudobnej teórie

Táto kapitola vysvetľuje základné pojmy z oblasti hudobnej teórie, ktoré sú pre pochopenie tejto práce dôležité. Objasňuje, čo tvorí melódiu a ako sa melódie zapisujú. Prvá sekcia sa venuje hudobným zvukom a ich vlastnostiam. Druhá sekcia ukazuje, ako sa tieto zvuky zapisujú. Tieto sekcie sa opierajú o poznatky z [5].

V poslednej sekcii je uvedený koncept hudobných foriem. Sekcia sa zameriava na predstavenie motívu, témy, motivickej práce alebo hudobných variácií. Táto sekcia sa opiera najmä o poznatky z [13] a [15].

2.1 Hudobné zvuky – tóny

V hudobnej teórii sú základným stavebným prvkom melódie tóny. Táto sekcia definuje, čo tóny sú a ako sa rozlišujú.

Zvuky vznikajú chvením pružného telesa. Zvuky môžeme rozdeliť na hudobné a nehudobné, pričom za hudobné zvuky považujeme zvuky, ktoré vznikajú pravidelným chvením zdroja zvuku. Zvuky s nepravidelným chvením nazývame šramoty. Tóny sú hudobnými zvukmi, ktoré majú nasledujúce štyri vlastnosti:

- výška
- dĺžka
- sila (alebo intenzita)
- farba

Sila je určená tým, do akej šírky sa pružné teleso chveje, popisuje to, ako potichu, či nahlas tón znie. *Farba* je určená zdrojom zvuku, teda nástrojom, ktorý zvuk tvorí. Farba tónu je vlastnosť, vďaka ktorej vieme iba sluchom rozlíšiť, či daný zvuk pochádza z nástroja ako husle, alebo klarinet, alebo je tvorený hlasom.

Pre pochopenie tejto práce sú najdôležitejšie vlastnosti tónu jeho výška a jeho dĺžka. Dĺžka tónu je určená tým, ako dlho sa zdroj zvuku chveje. Výška tónu je určená frekvenciou chvenia zdroja zvuku.

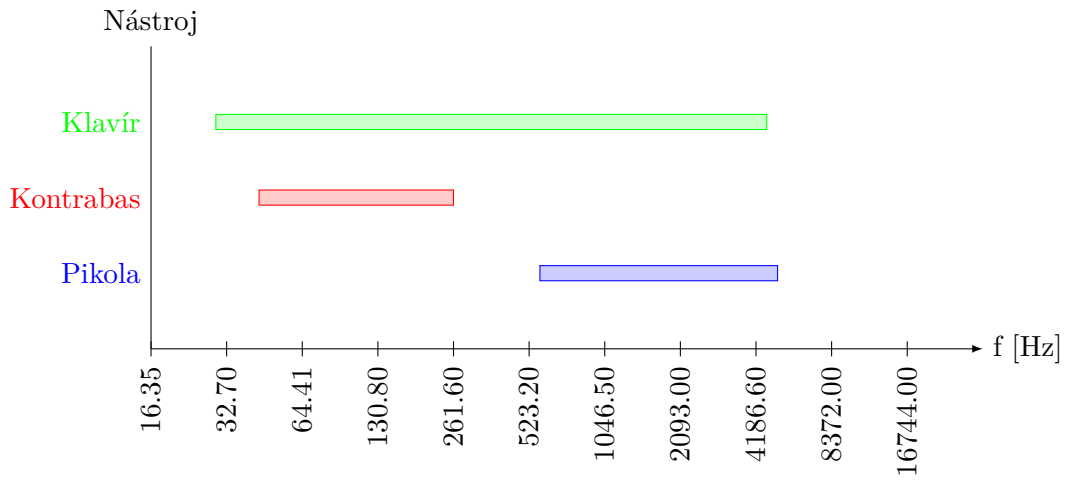
Frekvencia je fyzikálna veličina, ktorá vyjadruje počet kmitov za sekundu. Základnou jednotkou je hertz, pričom jeden hertz značí jeden kmit za sekundu. Čím viac kmitov za sekundu znejúce teleso vykoná, tým *vyššie* tón znie, a naopak, čím nižšia frekvencia, tým *nižšie* alebo *hlbšie* tón znie. Pre predstavu sú na obrázku 2.1 znázornené frekvenčné rozsahy vybraných hudobných nástrojov.

V hudbe sa väčšinou nepoužívajú zvuky všetkých frekvencii, ale frekvencie vzdialené v istých intervaloch. Pre pomenúvanie tónov sa využívajú písmená latinskej abecedy. Podľa nemeckej konvencie používame ako *hudobnú abecedu* rad mien siedmich základných tónov – c, d, e, f, g, a, h. V hudbe používame tónov viac než sedem, preto sa tento rad niekoľkokrát opakuje.

Aby sme každému tónu pridelili unikátne označenie, delíme celý súbor tónov do menších skupín zvaných oktávy (oktáva znamená ôsmy tón), ktoré medzi sebou rozlišujeme typom písma (veľké a malé) a indexovaním.

C_2	D_2	E_2	F_2	G_2	A_2	H_2	subkontra
C_1	D_1	E_1	F_1	G_1	A_1	H_1	kontra
C	D	E	F	G	A	H	veľká
c	d	e	f	g	a	h	malá
c^1	d^1	e^1	f^1	g^1	a^1	h^1	jednočiarkovaná
c^2	d^2	e^2	f^2	g^2	a^2	h^2	dvojčiarkovaná
c^3	d^3	e^3	f^3	g^3	a^3	h^3	trojčiarkovaná
c^4	d^4	e^4	f^4	g^4	a^4	h^4	štvorčiarkovaná

Tabuľka 2.1: Značenie tónov v jednotlivých oktávach. Tabuľka prevzatá z [5].



Obr. 2.1: Frekvenčné rozsahy vybraných hudobných nástrojov. Jednotlivé dieliky sú vzdialené vždy o oktávu. Rozsahy prevzaté z [14].

V európskej hudbe sa ustálila konvencia ladenia podľa tónu nazývaného *komorné a* s frekvenciou 440 Hz. Podľa neho sa v prevažnej väčšine ladia nástroje v orchestroch. Komorné *a* sa v nemeckej konvencii nazýva taktiež *jednočiarkované a* alebo *a jedna*. Vzdialenosť dvoch tónov alebo ich výškový rozdiel vyjadrujeme v intervaloch. *Oktáva* je interval, kde vyšší tón intervalu má dvojnásobnú frekvenciu oproti nižšiemu tónu intervalu. Napríklad pri zdvojnásobení frekvencie *jednočiarkovaného a* dostaneme hodnotu frekvencie *dvojčiarkovaného a*. Teda

$$a_1 = 440\,\text{Hz}$$

$$a_2 = 880\,\text{Hz}$$

Oktáva je rozdelená na 12 tzv. *poltónov* tak, aby frekvencie tónov vzdialených o poltón boli vždy v rovnakom pomere. Pre získanie frekvencie tónu o poltón vyššieho vynásobíme frekvenciu tónu konštantou $\sqrt[12]{2}$.

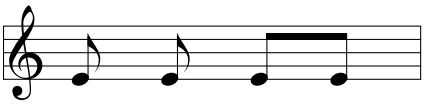
2.2 Zápis melódie – notové písmo

Nota je grafickou reprezentáciou tónu. Tvarom noty určujeme dĺžku tónu. Celá nota môže byť rozdelená na dve polovice, štyri štvrtiny, atď. Takto delíme celú notu na dve *polové*, štyri *štvrťové*, osem *osminových* a 16 *šestnástinových*. V tabuľke 2.2 sú znázornené tvary týchto nôt.

	celá
	polová
	štvrťová
	osminová
	šestnástinová

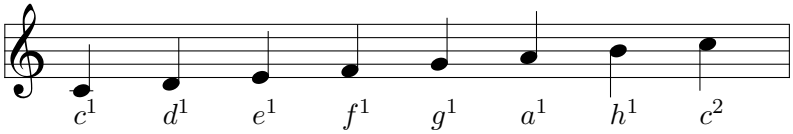
Tabuľka 2.2: Názvy nôt podľa dĺžky.

Nota má vždy *hlavičku*, ktorá môže byť vyplnená alebo prázdna. Ďalej nota môže mať *nožičku* a *zástavku*. Nožička sa píše na pravej strane smerom dohora a na ľavej strane hlavičky smerom dolu. Zástavky sa môžu nahradiť trámcom, ak za sebou nasleduje viac nôt so zástavkou ako na obrázku 2.2.



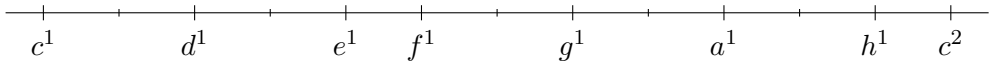
Obr. 2.2: Noty so zástavkami a noty spojené trámcom.

Noty sa zapisujú do notovej osnovy, ktorá sa skladá z piatich horizontálnych čiar a štyroch medzier. Notu môžeme umiestniť na čiaru alebo do medzery. Vertikálna pozícia noty (jej hlavičky) v notovej osnove reprezentuje výšku tónu. Čím vyššie je nota položená, tým vyšší tón reprezentuje. Pre zobrazenie nôt, ktoré sú mimo notovej osnovy, sa používajú tzv. *pomocné čiary*, ktoré môžu byť umiestnené nad alebo pod notovou osnovou (na obrázku 2.3 je nota c^1 na prvej pomocnej čiare pod notovou osnovou).

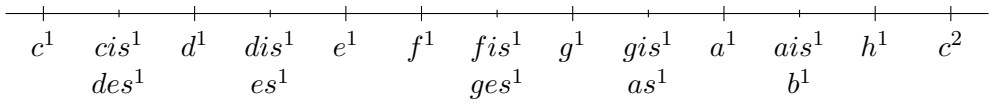


Obr. 2.3: Stupnica c-dur. Názvy nôt podľa výšky.

V skladbách sa väčšinou nepoužívajú všetky dostupné výšky tónov, ale melódie sa pohybujú v nejakej tónine. Tónina skladby je určená podľa stupnice, z ktorej obsahuje najviac tónov. Diatonická stupnica sa skladá z takých tónov, aby sa medzi nimi päťkrát vyskytovala vzdialenosť o celý tón a dvakrát o poltón. Diatonické stupnice delíme na durové a molové, kde v durových sa poltóny vyskytujú medzi tretím a štvrtým tónom a siedmym a ôsmym tónom stupnice a pri molových je to medzi druhým a tretím a piatym a šiestym. Na obrázku 2.3 sú poltóny medzi tónmi e^1 – f^1 a h^1 – c^2. Vzdialenosti tónov v stupnici c-dur sú zobrazené aj na obrázku 2.4, kde jeden dielik reprezentuje vzdialenosť o poltón.

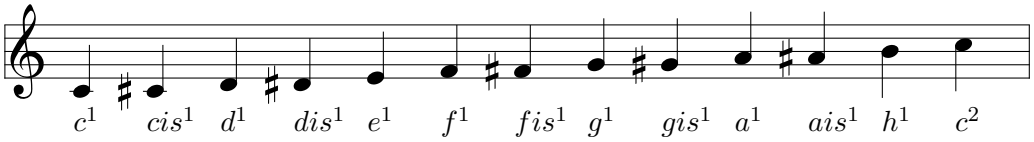


Obr. 2.4: Vzdialenosti v poltónoch medzi tónmi stupnice c-dur.



Obr. 2.5: Zobrazenie pomenovaní tónov mimo základnej hudobnej abecedy.

Pre značenie ostatných tónov, ktoré sú mimo základnej tónovej abecedy, sa používajú *posuvky* – krížiky a béčka, ktoré sa píšu pred notu, ktorej hodnotu menia. Krížik – ♯ notu zvyšuje o poltón a k názvu tónu pridáva príponu *-is* a béčko – ♭ notu o poltón znižuje a pridáva k názvu tónu príponu *-es*. V prípade zníženého *a* sa používa pomenovanie *as* a pri zníženom *h* sa namiesto *hes* nota nazýva *b*. Pomenovania týchto tónov sú zobrazené na obrázku 2.5. Posuvky majú platnosť do konca taktu. Na zrušenie posuvky sa používa *odrážka* – ♮ . Chromatická stupnica je taká, v ktorej sa vyskytujú všetky poltóny oktávy.

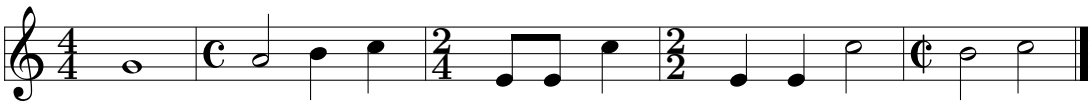


Obr. 2.6: Chromatická stupnica. Názvy nôt podľa výšky.

Bez použitia pomocných čiar je možné do notovej osnovy zapísať rozsah približne jednej oktávy, ale ako je vidieť na obrázku 2.1, rôzne nástroje sa môžu pohybovať v rôznych výškach. Preto boli zavedené symboly, ktoré určujú položenie nejakého tónu, ktorý je potom použitý ako referenčný bod pre zápis nôt do notovej osnovy. Tieto symboly sa nazývajú kľúče. Na obrázku 2.3 je prvým symbolom v notovej osnove *husľový kľúč*, ktorý sa začína písať od druhej čiary. Husľový kľúč určuje položenie tónu g^1, preto sa nazýva tiež *kľúč G*. Kľúčov existuje viac, ale nie sú pre túto prácu relevantné.

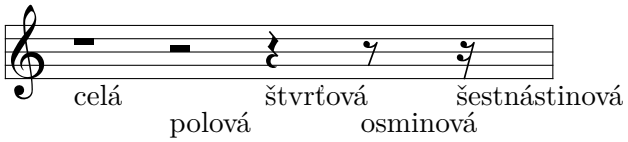
Hudbu sa pri zápise organizuje do menších jednotiek, ktoré sa nazývajú takty. Takty majú určitú dĺžku, ktorá sa počíta v dobách. Dĺžku taktu určuje taktové predznamenanie, ktoré sa skladá z dvoch čísel. Horné číslo určuje, koľko dôb je v takte a spodné číslo určuje, ktorá nota má trvanie jednej doby. Číselná reprezentácia noty je odvodená od toho, na koľko častí delí notu celú. Napríklad prvý takt na obrázku 2.7 značí štvorštvrťový takt, v jednom takte sú štyri doby a štvrťová nota má trvanie jednej doby. V praxi sa najčastejšie využívajú takty dvojštvrťový, trojštvrťový a štvorštvrťový a šesťosminový. Okrem nich sa

však v hudobných dielach nájdu aj menej tradičné takty ako napríklad päťštvrťový, sedemosminový a iné. Určenie taktu nasleduje za kľúčom. Koniec taktu sa značí vertikálnou čiarou a koniec skladby dvojitou vertikálnou čiarou. Na obrázku 2.7 vidno štvorštvrťový, dvojštvrťový a dvojpolový takt. Druhý takt ukazuje alternatívne značenie štvorštvrťového taktu a posledný takt je dvojpolový.



Obr. 2.7: Príklady taktových predznamenaní.

V melódii sa niekedy vyskytujú pauzy. Tie sa značia symbolmi, ktoré sa nazývajú pomlčky. Podobne ako pri notách, trvanie pomlčky je určené jej tvarom.

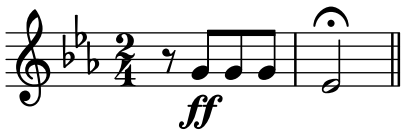


Obr. 2.8: Názvy pomlčiek.

2.3 Motív, téma a variácie

Táto sekcia predstaví, čo je motív a téma v hudbe a ako sa s nimi dá pracovať. Poznatky o hudobných formách a variačných technikách sú prevzaté z [13] a [15].

Motív je krátky, výrazný a pomerne ucelený hudobný úryvok, ktorého hlavnou zložkou je obyčajne melódia, tj. melodická linka a rytmus [15]. Motív má väčšinou dĺžku jedného až dvoch taktov. V skladbe sa motív zvykne opakovať bez zmeny alebo s malými obmenami. Existujú prípady, kedy tvorí motív iba rytmický útvar, napríklad úvod skladby *We will rock you* od skupiny Queen alebo tlieskanie v skladbe *Niech mówią, że to nie jest miłość* od skladateľa a dirigenta Piotra Rubika. Príkladom melodického motívu je krátky štvortónový motív z *V. symfónie* L. v. Beethovena.



Obr. 2.9: Motív z V. symfónie L. v. Beethovena [5].

Téma, podobne ako motív, predstavuje hudobnú myšlienku, ale oproti motívu je téma rozsiahlejším útvarom.

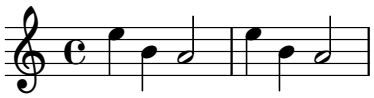
V hudbe rozlišujeme rôzne hudobné formy. Hudobné formy popisujú ako sú časti hudobného diela usporiadané a aké sú vzťahy medzi nimi. Pri tvorbe hudobnej skladby nejakej formy dodržiava skladateľ určité pravidlá. Analýzou týchto pravidiel je možné určiť formu hudobnej skladby. Pri skúmaní hudobných foriem sa jednotlivé časti skladby značia veľkými

písmenami latinskej abecedy. Príkladom hudobnej formy je malá trojdielna piesňová forma, ktorá má formu A B A.

Táto práca sa venuje variačnej hudobnej forme. Variačná hudobná forma, tiež nazývaná téma a variácie, predstavuje v muzikológii hudobnú formu, kedy sa nejaká hudobná pasáž alebo téma opakuje s rôznymi obmenami. Ak sa pre označenie témy použije písmeno A, potom sa pre značenie jej variácií použije A1 A2 a tak ďalej. Variačné techniky sa môžu uplatňovať ako na tému tak aj na motív. Ďalej budú uvedené jednoduché typy variácií.

Opakovanie

Opakovanie je najjednoduchším typom. Na obrázku 2.10 je ukázaný krátky motív a potom jeho opakovanie v nasledujúcom takte.

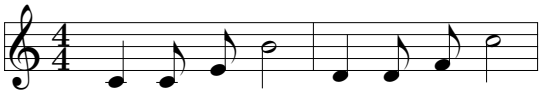


Obr. 2.10: Opakovanie motívu.

Sekvencia

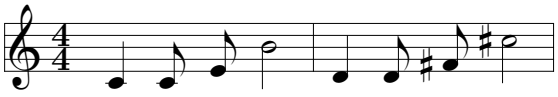
Sekvencie sú typy variácií, kedy sa melódia zopakuje výškovo posunutá o stupeň vyššie alebo nižšie. Dĺžky tónov ostávajú nezmenené. Rozlišujeme dva typy sekvencií:

- Tonálna sekvencia je posúvanie melódie v rámci diatonickej stupnice. To znamená, že nie každý tón bude posunutý o rovnaký interval.
- Reálna sekvencia je posunutie každého tónu melódie o rovnaký interval. V niektorých prípadoch, ale využíva tóny mimo pôvodnej tóniny.



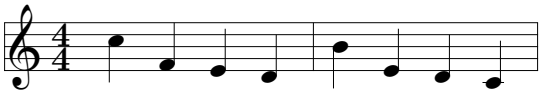
Obr. 2.11: Tonálna sekvencia smerom nahor.

Na obrázkoch 2.11 a 2.12 sú pre porovnanie ukážky tonálnej a reálnej sekvencie smerom nahor. Na obrázkoch je ukázané, že niektoré tóny sa pri reálnej sekvencii posúvajú rovnako ako pri tonálnej a niektoré tóny sa zvýšia na tóny mimo tóninu.

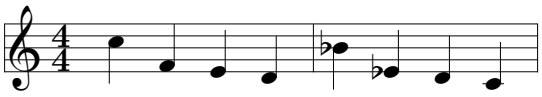


Obr. 2.12: Reálna sekvencia smerom nahor.

Na obrázkoch 2.13 a 2.14 sú pre porovnanie ukážky tonálnej a reálnej sekvencie toho istého motívu.



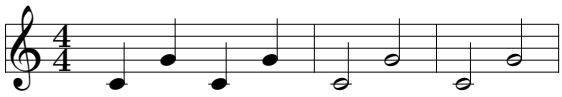
Obr. 2.13: Tonálna sekvencia smerom nadol.



Obr. 2.14: Reálna sekvencia smerom nadol.

Augmentácia

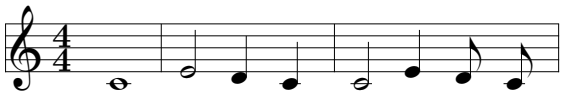
Pri augmentácii melódie výška tónov ostáva nezmenená. Bez zmeny tempa sa dĺžka tónov zdvojnásobí.



Obr. 2.15: Augmentácia motívu z prvého taktu.

Diminúcia

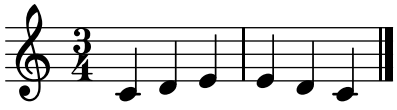
Diminúcia značí skrátenie dĺžky tónov melódie. Výšky tónov ostávajú nezmenené.



Obr. 2.16: Diminúcia motívu z prvých dvoch taktov.

Retrogradácia

Retrogradácia je hranie tónov témy odzadu, od posledného tónu k prvému. Výška aj dĺžka tónov ostáva nezmenená.



Obr. 2.17: Retrogradácia.

Kapitola 3

Formálne modely

Táto kapitola je zameraná na teóriu z oblasti formálnych modelov. V sekcii 3.1 sú predstavené množiny, ich vlastnosti a operácie nad nimi. Ďalej, v sekcii 3.2 budú vysvetlené symboly, abecedy a reťazce ako základné stavebné prvky jazyka. Sekcia 3.3 predstaví gramatiky a jazyky Chomského hierarchie.

Cieľom tejto kapitoly je najprv zoznámiť čitateľa s prostriedkami na porozumenie teórie formálnych jazykov a potom predstaviť formálne modely, ktoré jazyky definujú.

3.1 Množiny

V sekcii sú zhrnuté základné pojmy a poznatky z teórie množín. Definície sú prevzaté z [9].

Množina je kolekcia elementov, ktoré sú navzájom rozlíšiteľné. Ak množina M obsahuje element a, zapisujeme to ako $a \in M$ a hovoríme, že element a *patrí* do množiny M. Ak máme element a, ktorý nie je obsiahnutý v množine M, hovoríme, že element a *nepatrí* do množiny M a zapisujeme to ako $a \notin M$. *Kardinalita* alebo *mohutnosť* množiny M je počet elementov v M a značí sa $\text{card}(M)$. *Prázdna množina*, množina, ktorá neobsahuje žiadne elementy, sa značí $\emptyset$. Ak máme množinu $M = \emptyset$, potom $\text{card}(M) = 0$. Množina M je *konečná*, ak obsahuje konečný počet elementov, inak je množina *nekonečná*.

Konečnú množinu M môžeme špecifikovať vymenovaním jej elementov nasledovne

$$M = \{a_1, a_2, ..., a_n\},$$

kde elementy od a_1 po a_n sú členmi množiny M.

Príklad 3.1.1. Majme množinu M, ktorá obsahuje prirodzené čísla menšie ako 3. Množinu môžeme špecifikovať ako

$$M = \{1, 2\}.$$

O množine M platí napríklad $1 \in M$, $3 \notin M$ a $\text{card}(M) = 2$.

Nekonečnú množinu M zvyčajne špecifikujeme pomcou vlastnosti π tak, že množina M obsahuje všetky elementy, ktoré túto vlastnosť spĺňajú. Takáto špecifikácia má všeobecne tvar

$$M = \{a | \pi(a)\}.$$

Množinu z príkladu 3.1.1 môžeme definovať vlastnosťou nasledovne

$$M = \{a | a \text{ je prirodzené číslo a } a < 3\}$$

alebo

$$M = \{a | a \in \mathbb{N}, a < 3\}.$$

Množina M_1 je *podmnožinou* množiny M_2, ak pre každý element množiny M_1 platí, že patrí aj do množiny M_2, značíme ako $M_1 \subseteq M_2$. Ak pre množiny M_1 a M_2 platí $M_1 \subseteq M_2$ a zároveň množina M_2 obsahuje aspoň jeden element a, pre ktorý platí $a \notin M_1$, potom M_1 je *vlastná podmnožina* množiny M_2 a značíme ako $M_1 \subset M_2$. Množina M_1 *sa rovná* množine M_2, ak platí $M_1 \subseteq M_2$ a súčastne $M_2 \subseteq M_1$, značí sa $M_1 = M_2$.

Definícia 3.1.1. *Multimnožina* je zovšeobecnením množiny, ktoré povoľuje viacnásobný výskyt elementov.

Operácie nad množinami

Majme množiny M_1 a M_2. *Zjednotenie* množín M_1 a M_2 značíme $M_1 \cup M_2$ a je definované ako

$$M_1 \cup M_2 = \{x | x \in M_1 \text{ alebo } x \in M_2\}.$$

Prienik množín M_1 a M_2 značíme $M_1 \cap M_2$ a definujeme ako

$$M_1 \cap M_2 = \{x | x \in M_1 \text{ a súčasne } x \in M_2\}.$$

Rozdiel množín M_1 a M_2 značíme $M_1 - M_2$ a definujeme ako

$$M_1 - M_2 = \{x | x \in M_1 \text{ a súčasne } x \notin M_2\}.$$

3.2 Abeceda, reťazec

Definícia 3.2.1. *Abeceda* je konečná, neprázdna množina elementov, ktoré nazývame *symboly.*

Definícia 3.2.2. Nech Σ je abeceda. Potom ε označuje *prázdny reťazec*, reťazec, ktorý neobsahuje žiadne symboly. Ak x je *reťazec nad abecedou* Σ a $a \in \Sigma$, potom xa je tiež *reťazec nad abecedou* Σ.

Definícia 3.2.3. Nech x je reťazec nad abecedou Σ. *Dĺžka reťazca* x, sa značí $|x|$ a je definovaná ako počet symbolov v reťazci x.

1. ak $x = \varepsilon$, potom $|x| = 0$
2. ak $a_1 ... a_n$, pre nejaké $n \geq 1$, kde $a_i \in \Sigma$ pre všetky $i = 1, ..., n$, potom $|x| = n$.

Definícia 3.2.4. Nech x a y sú dva reťazce nad abecedou Σ. Potom reťazec xy je *konkatenácia* reťazcov x a y. Pre každý reťazec x platí

$$x\varepsilon = \varepsilon x = x$$

Príklad 3.2.1. Majme abecedu $\Sigma = \{a, b\}$. Podľa definície 3.2.2 vieme, že ε je reťazec nad Σ. A ďalej vieme, že konkatenáciou (definícia 3.2.4) reťazca nad Σ a symbolu patriaceho do Σ získame reťazec, ktorý je tiež reťazcom nad abecedou Σ. Z toho vyplýva, že a je reťazcom nad abecedou Σ a tiež b je reťazcom nad Σ.

Definícia 3.2.5. Nech x je reťazec nad abecedou Σ. Pre $i \geq 0$, i-ta *mocnina* x, x^i, je rekurzívne definovaná ako

1. $x^0 = \varepsilon$
2. $x^i = xx^{i-1}$, pre $i \geq 1$.

Príklad 3.2.2. Ukážka konkatenácie a mocniny reťazca. $a^3b^2 = aaabb$

Definícia 3.2.6. Nech x je reťazec nad abecedou Σ. *reversal(x)* je definovaný ako

1. ak $x = \varepsilon$, potom reversal$(x) = \varepsilon$
2. ak $x = a_1...a_n$, pre $n \geq 1$ a $a_i \in \Sigma$, pre $i = 1, ..., n$, potom reversal$(a_1...a_n) = a_n...a_1$.

Ak Σ značí abecedu, potom Σ^* značí množinu všetkých reťazcov, ktoré môžeme vytvoriť nad abecedou Σ. Σ^+ značí množinu všetkých reťazcov nad abecedou Σ okrem prázdneho reťazca, teda

$$\Sigma^+ = \Sigma^* - \{\varepsilon\}$$

Príklad 3.2.3. Nech $\Sigma = \{a\}$. Potom a, aa, aaa sú reťazce nad abecedou Σ.

Definícia 3.2.7. Majme abecedu Σ. Nech $L \subseteq \Sigma^*$. Potom L je *jazyk* nad abecedou Σ.

Definícia 3.2.8. Nech L je jazyk. L je *konečný* jazyk, ak card$(L) = n$, inak L je jazyk *nekonečný*.

Nad jazykmi sú definované množinové operácie *zjednotenie*, *prienik* a *rozdiel* ako v sekcii 3.1. Navyše definujeme doplnok jazyka nasledovne.

Definícia 3.2.9. Ak máme jazyk L nad abecedou Σ, potom *doplnok jazyka* L, značíme $\overline{L}$ a definujeme ako

$$\overline{L} = \Sigma^* - L$$

3.3 Gramatiky

Gramatiky boli zavedené ako nástroj, ktorý pomocou konečnej množiny pravidiel môže generovať aj nekonečný jazyk [3]. Táto kapitola uvádza definície gramatík a jazykov Chomského hierarchie. V [4] sú predstavené neobmedzené prepisovacie systémy ako gramatiky typu 0. Potom sú nad týmito systémami na prepisovacie pravidlá zavedené obmedzenia, ktoré sú postupne silnejšie. To znamená, že pre každé i, gramatika *typu i+1* spĺňa aj obmedzenie pre gramatiku *typu i*. V Chomského hierarchii rozlišujeme gramatiky typu 0, typu 1, typu 2 a typu 3, kde vyššie typy postupne definujú menej všeobecné gramatiky a jazyky.

Definícia 3.3.1. *Frázová* alebo *všeobecná gramatika* je štvorica $G = (N, T, P, S)$, kde

- N je abeceda neterminálov
- T je abeceda terminálov, pričom platí $N \cap T = \emptyset$
- P je konečná množina pravidiel v tvare $x \to y$, kde $x \in (N \cup T)^* N (N \cup T)^*$ a $y \in (N \cup T)^*$
- S je počiatočný symbol, pričom $S \in N$

Neterminály sú symboly, ktoré môžeme pomocou pravidiel prepisovať. *Terminály* sú symboly, ktoré sa ďalej neprepisujú. Zjednotenie abecedy terminálov a abecedy neterminálov definuje *úplnú abecedu* gramatiky [12]. Úplná abeceda sa zvykne značiť V podľa anglického *vocabulary* [4].

Definícia 3.3.2. Nech $G = (N, T, P, S)$ je frázová gramatika, $p = x \to y \in P$, a $u, v \in (N \cup T)^*$. Potom *priama derivácia* alebo *derivačný krok* je binárna relácia reťazcov nad $(N \cup T)^*$, značí sa $\Rightarrow_G$ a je definovaná ako

$$A \Rightarrow_G B[p],$$

ak $A = uxv, B = uyv$.

Hovoríme, že A priamo derivuje B podľa p v gramatike G alebo gramatika G robí derivačný krok z A do B podľa p [12]. Skrátene môžeme značiť ako $A \Rightarrow_G B$ bez uvedenia použitého pravidla. Reflexívny tranzitívny uzáver relácie $\Rightarrow_G$ značíme $\Rightarrow_G^*$.

Definícia 3.3.3. *Jazyk* generovnaý gramatikou G značíme $L(G)$ a je definovaný ako

$$L(G) = \{w | w \in T^*, S \Rightarrow_G^* w\}$$

Generovaný jazyk je teda množina takých reťazcov nad T^*, ktoré vieme získať postupnosťou derivačných krokov z S do w podľa G [12].

Frázové gramatiky v Chomského hierarchii patria pod typ 0. Jazyk typu 0 je taký, ktorý je generovaný gramatikou typu 0. Frázové gramatiky generujú rodinu rekurzívne spočítateľných jazykov, ktorú značíme RE (z anglického *recursively enumerable*) a sú ekvivalentné s rodinou jazykov typu 0. Jazyky typu 0 sú prijímané Turingovým strojom [12].

Definícia 3.3.4. *Kontextová gramatika* je frázová gramatika $G = (N, T, P, S)$, kde každé pravidlo $x \to y \in P$ je v tvare

$$u_1 A u_2 \to u_1 a u_2,$$

kde $u_1, u_2 \in (N \cup T)^*$, $A \in N$ a $a \in (N \cup T)^+$.

Kontextové gramatiky generujú kontextové jazyky a v Chomského hierarchii patria pod typ 1. Rodinu kontextových jazykov značíme CS (z anglického *context-sensitive*) [12]. Rodina kontextových jazykov je podmnožina rodiny rekurzívne spočítateľných jazykov.

$$CS \subset RE$$

Kontextové gramatiky sú ekvivalentné lineárne ohraničeným automatom [9].

Definícia 3.3.5. *Bezkontextová gramatika* je frázová gramatika $G = (N, T, P, S)$, kde pravidlá majú tvar

$$A \to a,$$

kde $A \in N$ a $a \in (N \cup T)^*$ [10].

Bezkontextové gramatiky generujú bezkontextový jazyk a v Chomského hierarchii patria pod typ 2. Ekvivalentom bezkontextových gramatík sú zásobníkové automaty. Obmedzenie oproti gramatikám typu 1 spočíva v obmedzení ľavej strany produčných pravidiel tak, aby obsahovali práve 1 neterminál.

Rodina jazykov, ktoré sú generované bezkontextovými gramatikami, sa značí CF (podľa anglického *context-free*) a je podmnožinou rodiny kontextových jazykov.

$$CF \subset CS \subset RE$$

Definícia 3.3.6. Frázová gramatika $G = (N, T, P, S)$ sa nazýva *lineárna gramatika*, ak pre každé pravidlo

$$x \to y \in P$$

platí

$$x \in N, y \in T^* \cup T^*NT^* \text{ [12]}.$$

To znamená, že ľavá strana pravidla obsahuje práve jeden neterminál a pravá strana pravidla je reťazec, ktorý obsahuje 0 až n terminálov a môže obsahovať najviac jeden neterminál.

Lineárne gramatiky generujú *lineárne jazyky*. Rodina lineárnych jazykov sa označuje *LIN*.

Lineárne gramatiky môžeme viac špecializovať zavedením obmedzenia na pravú stranu pravidla tak, že ak obsahuje neterminál, bude umiestnený iba na začiatku alebo na konci reťazca. Ak pre pravidlá $x \to y \in P$ platí $x \in N, y \in T^* \cup T^*N$ hovoríme o *pravo-lineárnej gramatike*. Ak pre pravidlá platí $x \in N, y \in T^* \cup NT^*$ hovoríme o *ľavo-lineárnej gramatike*. Pravo-lineárne gramatiky generujú *pravo-lineárne jazyky*, ktoré označujeme *RLIN* (podľa anglického *right-linear*) [10].

Rodina jazykov, ktoré sú generované pravo-lineárnymi a ľavo-lineárnymi gramatikami sú ekvivalentné navzájom a tiež sú ekvivalentné s rodinou jazykov, ktoré sú generované regulárnymi gramatikami. V Chomského hierarchii patria pod typ 3 [12].

Definícia 3.3.7. *Regulárna gramatika* je frázová gramatika $G = (N, T, P, S)$, ak pre každé pravidlo

$$x \to y \in P$$

platí

$$x \in N, y \in T \cup TN \cup \{\varepsilon\} \text{ [12]}.$$

Jazyk generovaný regulárnou gramatikou nazývame *regulárny jazyk*. Rodinu regulárnych jazykov značíme *REG*.

$$REG = RLIN \subset LIN \subset CF \subset CS \subset RE$$

3.4 Automaty

V predchádzajúcej sekcii boli predstavené gramatiky ako nástroje pre generovanie reťazcov jazyka. Automaty sú prostriedkami, ktoré prijímajú reťazce jazyka. V Chomského hierarchii sú typom jazykov priradené gramatiky, ktoré daný typ generujú. Ako bolo spomenuté v predchádzajúcej sekcii, jednotlivým typom sú priradené aj iné modely, ktoré jazyky prijímajú. Táto sekcia predstaví niektoré z týchto modelov.

Konečné automaty patria v Chomského hierarchii pod typ 3.

Definícia 3.4.1. Konečný automat je pätica

$$M = (Q, \Sigma, R, s, F),$$

kde

- Q je konečná množina stavov,
- Σ je vstupná abeceda, pre ktorú platí $\Sigma \cap Q = \emptyset$,

- R je konečná množina pravidiel v tvare $pa \to q$, kde $p, q \in Q$ a $a \in \Sigma \cup \{\varepsilon\}$,
- $s \in Q$ je počiatočný stav,
- $F \subseteq Q$ je množia koncových stavov [9].

Zásobníkové automaty majú rovnakú silu ako bezkontextové gramatiky. V Chomského hierarchii patria pod typ 2.

Definícia 3.4.2. Zásobníkový automat je pätica

$$M = (Q, \Sigma, R, s, F),$$

kde

- Q je konečná množina stavov,
- Σ je abeceda, pre ktorú platí $\Sigma \cap Q = \emptyset$ a $\Sigma = \Sigma_I \cup \Sigma_{PD}$, kde Σ_I značí vstupnú abecedu a Σ_{PD} značí zásobníkovú abecedu, ktorá obsahuje počiatočný symbol na zásobníku S,
- R je konečná množina pravidiel tvaru $Apa \to wq$, kde $A \in \Sigma_{PD}$, $p, q \in Q$, $a \in \Sigma_I \cup \{\varepsilon\}$ a $w \in \Sigma_{PD}^*$
- $s \in Q$ je počiatočný stav,
- $F \subseteq Q$ je množina koncových stavov [9].

Kapitola 4

Návrh

V tejto kapitole je navrhnutý model pre generáciu variácií podľa témy. V prvej sekcii je uvedená definícia gramatiky s rozptýleným kontextom, na ktorej je model založený. Potom sú v sekcii postupne zadefinované konkrétne abecedy a pravidlá modelu. V druhej sekcii je prezentovaný návrh algoritmu, ktorý bude použitý na generáciu reťazcov podľa vstupu.

4.1 Návrh modelu

Sekcia uvádza formálnu definíciu gramatík s rozptýleným kontextom a potom zadefinuje model gramatiky, ktorá bude použitá pre generovanie variácií. Gramatiky s rozptýleným kontextom boli zavedené z nutnosti definovania modelu, ktorý dáva do súvisu vzdialené časti vety. Takýto model dokáže zaviesť kontext medzi symbolmi, ktoré narozdiel od kontextových gramatík, môžu byť ľubovoľne vzdialené [7].

Definícia gramatiky s rozptýleným kontextom je prevzatá z [10].

Definícia 4.1.1. *Gramatika s rozptýleným kontextom* je štvorica $G = (V, T, P, S)$, kde

- V je úplná abeceda
- $T \subset V$ je abeceda terminálov
- P je konečná množina pravidiel v tvare

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

  kde $n \geq 1, A_i \in V - T$, a $x_i \in V^*$, pre všetky $i, 1 \leq i \leq n$ a každé pravidlo môže mať iné n
- $S \in V - T$ je počiatočný symbol

Ak

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

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

a $p = (A_1, ..., A_n) \rightarrow (x_1, ..., x_n) \in P$, kde $u_i \in V^*$, pre všetky $i, 1 \leq i \leq n + 1$, potom G robí derivačný krok z u do v podľa p a značí sa

$$u \Rightarrow_G v[p]$$

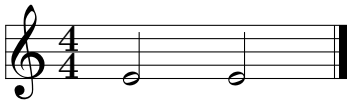
alebo skrátene ako $u \Rightarrow_G v$. Dĺžku pravidla p definujeme ako $\text{len}(p) = n$. Ak $\text{len}(p) \geq 2$, tak hovoríme, že p je kontextové pravidlo, inak pre $\text{len}(p) = 1$ hovoríme o pravidle bezkontextovom. *Jazyk* generovaný G značíme $L(G)$ a je definovaný ako

$$L(G) = \{w | w \in T^*, S \Rightarrow_G^* w\}$$

Jazyk L je *jazyk s rozptýleným kontextom*, ak existuje gramatika s rozptýleným kontextom G taká, že $L = L(G)$.

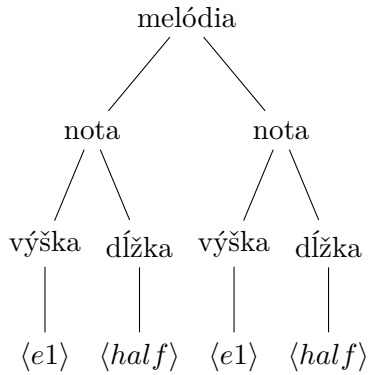
Gramatiky s rozptýleným kontextom sú gramatiky, ktoré prepisujú neterminály na reťazce terminálov a neterminálov. Prepisované neterminály môžu byť v rámci reťazca od seba ľubovoľne vzdialené a v rámci jedného derivačného kroku je tak možné vygenerovať časť témy a zároveň časť variácie, ktorá je na téme závislá.

Abeceda terminálov

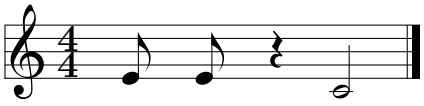


Obr. 4.1: Ukážka jednoduchej melódie.

Množinu terminálov budú tvoriť symboly reprezentujúce noty a pomlčky. Zvolené variácie upravujú vždy iba jednu vlastnosť tónu. Buď menia jeho výšku, alebo jeho dĺžku. Preto zvolená abeceda bude obsahovať symboly, ktoré predstavujú jedno alebo druhé. Zvolená abeceda má rozsah od c^1 po c^3. Výšky tónov budú reprezentované malým písmenom a číslom, napríklad c^1 ako $\langle$`c1`$\rangle$. Pomlčka bude značená ako $\langle$`rest`$\rangle$ a dĺžky budú reprezentované pomocou $\langle$`whole`$\rangle$, $\langle$`half`$\rangle$, $\langle$`quarter`$\rangle$ a $\langle$`eighth`$\rangle$ pre celú, polovú, štvrťovú a osminovú notu v tomto poradí. Nasledujúci graf ukazuje, ako je navrhnutá reprezentácia melódie, reprezentuje meódiu z obrázku 4.1.



Napríklad nasledujúci úryvok



bude reprezentovaný ako

$$\langle \texttt{e1} \rangle \langle \texttt{eighth} \rangle \langle \texttt{e1} \rangle \langle \texttt{eighth} \rangle \langle \texttt{rest} \rangle \langle \texttt{quarter} \rangle \langle \texttt{c1} \rangle \langle \texttt{half} \rangle$$

Dvojice symbolov reprezentujúce tón alebo pomlčku budú vždy usporiadané tak, aby prvý z nich reprezentoval výšku tónu alebo pomlčku a druhý jeho dĺžku. Umožní to vytvoriť oveľa menšiu množinu pravidiel než pri reprezentácii každej kombinácie výšky a dĺžky jedným symbolom.

$$\begin{aligned} T = &\{c1, d1, e1, f1, g1, a1, h1, c2, d2, e2, f2, g2, a2, h2, c3\} \cup \\ &\{whole, half, quarter, eighth\} \cup \\ &\{rest\} \end{aligned}$$

Abeceda neterminálov

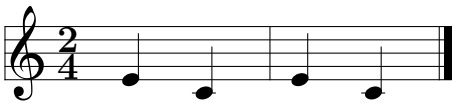
Tým, že reprezentácia jedného hudobného symbolu je rozdelená medzi dva terminály, na jej vyjadrenie použijeme dva neterminály. Ako štartovací neterminál sa použije $\langle \texttt{S} \rangle$. Pre vyjadrenie potreby striedania terminálov definujúcich výšku a terminálov definujúcich dĺžku sa použijú v pravidlách neterminály $\langle \texttt{X} \rangle$ a $\langle \texttt{Y} \rangle$.

$$N = \{S, X, Y\}$$

Pravidlá

Táto podsekcia na krátkych príkladoch postupne ukáže, ako boli zvolené pravidlá pre generáciu variácie podľa vstupu. Najprv si ukážeme tvorbu pravidiel pre variáciu opakovaním.

Opakovanie melódie



Obr. 4.2: Motív s opakovaním.

Na obrázku 4.2 je možné vidieť krátky motív v prvom takte a jeho opakovanie v druhom takte. Nemení sa výška ani dĺžka tónov, preto sú pre opakovanie motívu navrhnuté kopírovacie pravidlá. Pre kopírovanie dĺžky budú použité nasledujúce pravidlá.

$$\begin{aligned} &\{(Y, Y) \to (whole\ X, whole\ X), && (Y, Y) \to (half\ X, half\ X), \\ &\ (Y, Y) \to (quarter\ X, quarter\ X), && (Y, Y) \to (eighth\ X, eighth\ X)\} \subset P \end{aligned}$$

Pre kopírovanie výšky tónu boli navrhnuté nasledujúce pravidlá.

$$\begin{aligned} &\{(X, X) \to (c1\ Y, c1\ Y), && (X, X) \to (d1\ Y, d1\ Y), && (X, X) \to (e1\ Y, e1\ Y), \\ &\ (X, X) \to (f1\ Y, f1\ Y), && \dots\ , && (X, X) \to (g2\ Y, g2\ Y), \\ &\ (X, X) \to (a2\ Y, a2\ Y), && (X, X) \to (h2\ Y, h2\ Y), && (X, X) \to (c3\ Y, c3\ Y)\} \subset P \end{aligned}$$

Nasledujúce pravidlá

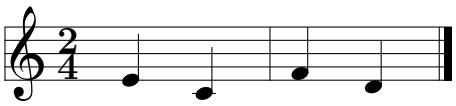
$$\begin{array}{lll} \{(S) \to (XX) & (X,X) \to (c1\,Y, c1\,Y), & (X,X) \to (e1\,Y, e1\,Y), \\ (Y,Y) \to (quarter\,X, quarter\,X), & (X,X) \to (\varepsilon,\varepsilon)\} \subset P & \end{array}$$

generujú melódie na obrázku 4.2 pomocou nasledujúcich derivácií.

$$\begin{aligned} \langle S\rangle &\Rightarrow_G \langle X\rangle\langle X\rangle \\ &\Rightarrow_G \langle e1\rangle\langle Y\rangle\langle e1\rangle\langle Y\rangle \\ &\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle X\rangle\langle e1\rangle\langle quarter\rangle\langle X\rangle \\ &\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle Y\rangle\langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle Y\rangle \\ &\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle\langle X\rangle\langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle\langle X\rangle \\ &\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle\langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle \end{aligned}$$

Tonálna sekvencia

Tonálna sekvencia opakuje melódiu o tón vyššie alebo nižšie pričom zachováva tóninu. Vstupnú abecedu tvoria výšky tónov v tónine C-dur, a tak aj pravidlá pre tonálne sekvencie budú obsahovať iba tóny tejto tóniny. Pri sekvencii sa dĺžka tónov nemení, preto budú použité kopírovacie pravidlá navrhnuté pre opakovanie témy.



Obr. 4.3: Posunutie motívu o tón vyššie.

Pre tonálnu sekvenciu nahor boli navrhnuté pravidlá

$$\begin{array}{lll} \{(X,X) \to (c1\,Y, d1\,Y), & (X,X) \to (d1\,Y, e1\,Y), & (X,X) \to (e1\,Y, f1\,Y)\} \\ (X,X) \to (f1\,Y, g1\,Y), & \dots, & (X,X) \to (g2\,Y, a2\,Y), \\ (X,X) \to (a2\,Y, h2\,Y), & (X,X) \to (h2\,Y, c3\,Y), & (X,X) \to (c3\,Y, d2\,Y) \subset P, \end{array}$$

kde v druhých komponentoch pravidla sú terminály výšky tónov o jeden stupeň vyššie oproti terminálom v prvých komponentoch. Tón c^3, ktorý je na kraji zvoleného rozsahu by v tonálnej sekvencii mal byť d^3, ale ten je už mimo zvoleného rozsahu. Ako riešenie je zvolená transpozícia tónu $d3$ o oktávu nižšie a tak namiesto pravidla

$$(X,X) \to (c3\,Y, d3\,Y)$$

je navrhnuté pravidlo

$$(X,X) \to (c3\,Y, d2\,Y).$$

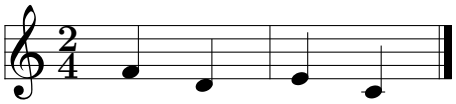
Nasledujúce pravidlá

$$\begin{array}{lll} \{(S) \to (XX), & (X,X) \to (c1\,Y, d1\,Y), & (X,X) \to (e1\,Y, f1\,Y), \\ (Y,Y) \to (quarter\,X, quarter\,X), & (X,X) \to (\varepsilon,\varepsilon)\} \subset P & \end{array}$$

generujú melódie na obrázku 4.3 pomocou nasledujúcich derivácií.

$$\begin{aligned}
\langle S\rangle &\Rightarrow_G \langle X\rangle\langle X\rangle \\
&\Rightarrow_G \langle e1\rangle\langle Y\rangle\langle f1\rangle\langle Y\rangle \\
&\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle X\rangle\langle f1\rangle\langle quarter\rangle\langle X\rangle \\
&\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle Y\rangle\langle f1\rangle\langle quarter\rangle\langle d1\rangle\langle Y\rangle \\
&\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle\langle X\rangle\langle f1\rangle\langle quarter\rangle\langle d1\rangle\langle quarter\rangle\langle X\rangle \\
&\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle\langle f1\rangle\langle quarter\rangle\langle d1\rangle\langle quarter\rangle
\end{aligned}$$

V tonálnej sekvencii smerom nadol posúvame výšku tónu v rámci tóniny o stupeň nižšie. Narazíme tu na podobný problém ako pri tonálnej sekvencii smerom nahor, avšak na opačnej strane zvoleného intervalu.



Obr. 4.4: Posunutie motívu o tón nižšie.

Tón o jeden stupeň nižší od tónu c^1 je malé h. Tento tón je však mimo zvoleného rozsahu. Ako riešenie je zvolený tón transponovaný o oktávu nahor tak, aby patril do zvoleného rozsahu. Preto je namiesto pravidla

$$(X, X) \to (c1\ Y, h\ Y)$$

navrhnuté pravidlo

$$(X, X) \to (c1\ Y, h1\ Y).$$

Pravidlá navrhnuté pre tonálnu sekvenciu nahor sú nasledujúce.

$$\begin{aligned}
\{&(X, X) \to (c1\ Y, h1\ Y), \quad (X, X) \to (d1\ Y, c1\ Y), \quad (X, X) \to (e1\ Y, d1\ Y), \\
&(X, X) \to (f1\ Y, e1\ Y), \quad \dots, \quad (X, X) \to (g2\ Y, f2\ Y), \\
&(X, X) \to (a2\ Y, g2\ Y), \quad (X, X) \to (h2\ Y, a2\ Y), \quad (X, X) \to (c3\ Y, h2\ Y)\} \subset P
\end{aligned}$$

Potom melódiu z obrázka 4.4 generujú pravidlá

$$\begin{aligned}
\{&(S) \to (XX), \quad (X, X) \to (d1\ Y, c1\ Y), \quad (X, X) \to (f1\ Y, e1\ Y), \\
&(Y, Y) \to (quarter\ X, quarter\ X), \quad (X, X) \to (\varepsilon, \varepsilon)\} \subset P
\end{aligned}$$

nasledujúcimi derivačnými krokmi.

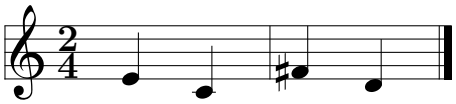
$$\begin{aligned}
\langle S\rangle &\Rightarrow_G \langle X\rangle\langle X\rangle \\
&\Rightarrow_G \langle f1\rangle\langle Y\rangle\langle e1\rangle\langle Y\rangle \\
&\Rightarrow_G \langle f1\rangle\langle quarter\rangle\langle X\rangle\langle e1\rangle\langle quarter\rangle\langle X\rangle \\
&\Rightarrow_G \langle f1\rangle\langle quarter\rangle\langle d1\rangle\langle Y\rangle\langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle Y\rangle \\
&\Rightarrow_G \langle f1\rangle\langle quarter\rangle\langle d1\rangle\langle quarter\rangle\langle X\rangle\langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle\langle X\rangle \\
&\Rightarrow_G \langle f1\rangle\langle quarter\rangle\langle d1\rangle\langle quarter\rangle\langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle
\end{aligned}$$

Reálna sekvencia

Reálna sekvencia je opakovanie melódie posunutej presne o celý tón vyššie alebo nižšie. Tým môžu vzniknúť variácie, ktoré budú obsahovať aj tóny mimo pôvodnej tóniny. V sekcii 2.2 bolo ukázané, že v stupnici C-dur sú poltóny medzi $e - f$ a $h - c$. Pre tieto tóny bude potrebné definovať nové pravidlá a terminály variácie. Pre jednoduchosť budeme používať iba značenie tónov pomocou krížikov. Pre variácie ostatných tónov je možné použiť pravidlá navrhnuté pre tonálne sekvencie. Pre dĺžky tónov budú použité kopírovacie pravidlá navrhnuté pre opakovanie témy.

Pri reálnej sekvencií smerom nahor sú tak navrhnuté nové pravidlá pre e^1, h^1, e^2 a h^2. Ak sa h^2 posunie o celý tón vyššie, výsledkom je tón cis^3, ten je však mimo zvoleného rozsahu. Tento problém je riešený ako v predchádzajúcich príkladoch. Pre reálnu sekvenciu smerom nahor sú zavedené nasledujúce nové pravidlá.

$$\{(X,X) \to (e1\,Y, fis1\,Y), \quad (X,X) \to (h1\,Y, cis2\,Y),$$
$$(X,X) \to (e2\,Y, fis2\,Y), \quad (X,X) \to (h2\,Y, cis2\,Y)\} \subset P$$



Obr. 4.5: Posunutie motívu o celý tón vyššie.

Melódiu na obrázku 4.5 je možné pomocou pravidiel

$$\{(S) \to (XX), \quad (X,X) \to (c1\,Y, d1\,Y), \quad (X,X) \to (e1\,Y, fis1\,Y),$$
$$(Y,Y) \to (quarter\,X, quarter\,X), \quad (X,X) \to (\varepsilon, \varepsilon)\} \subset P$$

zapísať nasledujúcimi derivačnými krokmi.

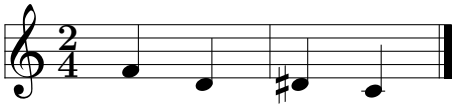
$$\begin{aligned}
\langle S\rangle &\Rightarrow_G \langle X\rangle\langle X\rangle \\
&\Rightarrow_G \langle e1\rangle\langle Y\rangle\langle fis1\rangle\langle Y\rangle \\
&\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle X\rangle\langle fis1\rangle\langle quarter\rangle\langle X\rangle \\
&\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle Y\rangle\langle fis1\rangle\langle quarter\rangle\langle d1\rangle\langle Y\rangle \\
&\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle\langle X\rangle\langle fis1\rangle\langle quarter\rangle\langle d1\rangle\langle quarter\rangle\langle X\rangle \\
&\Rightarrow_G \langle e1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle\langle fis1\rangle\langle quarter\rangle\langle d1\rangle\langle quarter\rangle
\end{aligned}$$

Pri reálnej sekvencii smerom nadol je potrebné definovať nové pravidlá pre tóny c^1, f^1, c^2, f^2 a c^3. Podobne ako v predchádzajúcich prípadoch, posunutím tónu c^1 sa dostávame mimo zvolený tónový rozsah. Pre reálnu sekvenciu sú navrhnuté nasledujúce pravidlá.

$$\{(X,X) \to (c1\,Y, ais1\,Y), \quad (X,X) \to (f1\,Y, dis1\,Y), \quad (X,X) \to (c2\,Y, ais1\,Y),$$
$$(X,X) \to (f2\,Y, dis2\,Y), \quad (X,X) \to (c3\,Y, ais2\,Y)\} \subset P$$

Melódiu na obrázku 4.6 je možné pomocou pravidiel

$$\{(S) \to (XX), \quad (X,X) \to (d1\,Y, c1\,Y), \quad (X,X) \to (f1\,Y, dis1\,Y),$$
$$(Y,Y) \to (quarter\,X, quarter\,X), \quad (X,X) \to (\varepsilon, \varepsilon)\} \subset P$$



Obr. 4.6: Posunutie motívu o celý tón nižšie.

zapísať nasledujúcimi derivačnými krokmi.

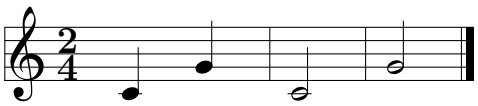
$$
\begin{aligned}
\langle S\rangle &\Rightarrow_G \langle X\rangle\langle X\rangle \\
&\Rightarrow_G \langle f1\rangle\langle Y\rangle\langle dis1\rangle\langle Y\rangle \\
&\Rightarrow_G \langle f1\rangle\langle quarter\rangle\langle X\rangle\langle dis1\rangle\langle quarter\rangle\langle X\rangle \\
&\Rightarrow_G \langle f1\rangle\langle quarter\rangle\langle d1\rangle\langle Y\rangle\langle dis1\rangle\langle quarter\rangle\langle c1\rangle\langle Y\rangle \\
&\Rightarrow_G \langle f1\rangle\langle quarter\rangle\langle d1\rangle\langle quarter\rangle\langle X\rangle\langle dis1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle\langle X\rangle \\
&\Rightarrow_G \langle f1\rangle\langle quarter\rangle\langle d1\rangle\langle quarter\rangle\langle dis1\rangle\langle quarter\rangle\langle c1\rangle\langle quarter\rangle
\end{aligned}
$$

Augmentácia

Augmentácia je opakovanie tónov motívu, ale s dvojnásobnou dĺžkou. Výška tónov sa pri augmentácii nemení, preto budú pre výšky tónov použité kopírovacie pravidlá navrhnuté pre opakovanie. Pre augmentáciu sú navrhnuté nasledujúce pravidlá.

$$
\begin{aligned}
&\{(Y,Y) \to (half\ X, whole\ X), && (Y,Y) \to (quarter\ X, half\ X), \\
&\ (Y,Y) \to (eighth\ X, quarter\ X)\} \subset P
\end{aligned}
$$

Pre celú notu bude použité kopírovacie pravidlo.



Obr. 4.7: Augmentácia motívu.

Melódiu na obrázku 4.7 je možné pomocou pravidiel

$$
\begin{aligned}
&\{(S) \to (XX), && (X,X) \to (c1\ Y, c1\ Y), && (X,X) \to (g1\ Y, g1\ Y), \\
&(Y,Y) \to (quarter\ X, half\ X), && (X,X) \to (\varepsilon, \varepsilon)\} \subset P
\end{aligned}
$$

zapísať nasledujúcimi derivačnými krokmi.

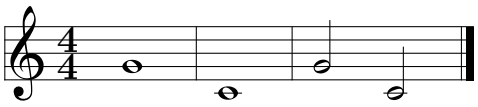
$$\begin{aligned}\langle S\rangle &\Rightarrow_G \langle X\rangle\langle X\rangle\\ &\Rightarrow_G \langle c1\rangle\langle Y\rangle\langle c1\rangle\langle Y\rangle\\ &\Rightarrow_G \langle c1\rangle\langle quarter\rangle\langle X\rangle\langle c1\rangle\langle half\rangle\langle X\rangle\\ &\Rightarrow_G \langle c1\rangle\langle quarter\rangle\langle g1\rangle\langle Y\rangle\langle c1\rangle\langle half\rangle\langle g1\rangle\langle Y\rangle\\ &\Rightarrow_G \langle c1\rangle\langle quarter\rangle\langle g1\rangle\langle quarter\rangle\langle X\rangle\langle c1\rangle\langle half\rangle\langle g1\rangle\langle half\rangle\langle X\rangle\\ &\Rightarrow_G \langle c1\rangle\langle quarter\rangle\langle g1\rangle\langle quarter\rangle\langle c1\rangle\langle half\rangle\langle g1\rangle\langle half\rangle\end{aligned}$$

Diminúcia

Diminúcia je opakovanie tónov motívu, ale s polovičnou dĺžkou. Výška tónov sa rovnako ako pri augmentácii nemení, preto budú pre výšky tónov znovu použité kopírovacie pravidlá navrhnuté pre opakovanie. Pre diminúciu sú navrhnuté nasledujúce pravidlá.

$$\{(Y,Y) \to (whole\ X, half\ X), \qquad (Y,Y) \to (half\ X, quarter\ X),$$
$$(Y,Y) \to (quarter\ X, eighth\ X)\} \subset P$$

Pre osminovú notu bude použité kopírovacie pravidlo.



Obr. 4.8: Diminúcia motívu.

Melódiu na obrázku 4.8 je možné pomocou pravidiel

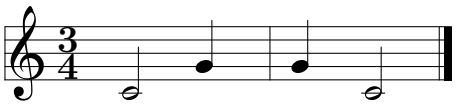
$$\{(S) \to (XX), \qquad (X,X) \to (c1\ Y, c1\ Y), \qquad (X,X) \to (g1\ Y, g1\ Y),$$
$$(Y,Y) \to (whole\ X, half\ X), \qquad (X,X) \to (\varepsilon,\varepsilon)\} \subset P$$

zapísať nasledujúcimi derivačnými krokmi.

$$\begin{aligned}\langle S\rangle &\Rightarrow_G \langle X\rangle\langle X\rangle\\ &\Rightarrow_G \langle g1\rangle\langle Y\rangle\langle g1\rangle\langle Y\rangle\\ &\Rightarrow_G \langle g1\rangle\langle whole\rangle\langle X\rangle\langle g1\rangle\langle half\rangle\langle X\rangle\\ &\Rightarrow_G \langle g1\rangle\langle whole\rangle\langle c1\rangle\langle Y\rangle\langle g1\rangle\langle half\rangle\langle c1\rangle\langle Y\rangle\\ &\Rightarrow_G \langle g1\rangle\langle whole\rangle\langle c1\rangle\langle whole\rangle\langle X\rangle\langle g1\rangle\langle half\rangle\langle c1\rangle\langle half\rangle\langle X\rangle\\ &\Rightarrow_G \langle g1\rangle\langle whole\rangle\langle c1\rangle\langle whole\rangle\langle g1\rangle\langle half\rangle\langle c1\rangle\langle half\rangle\end{aligned}$$

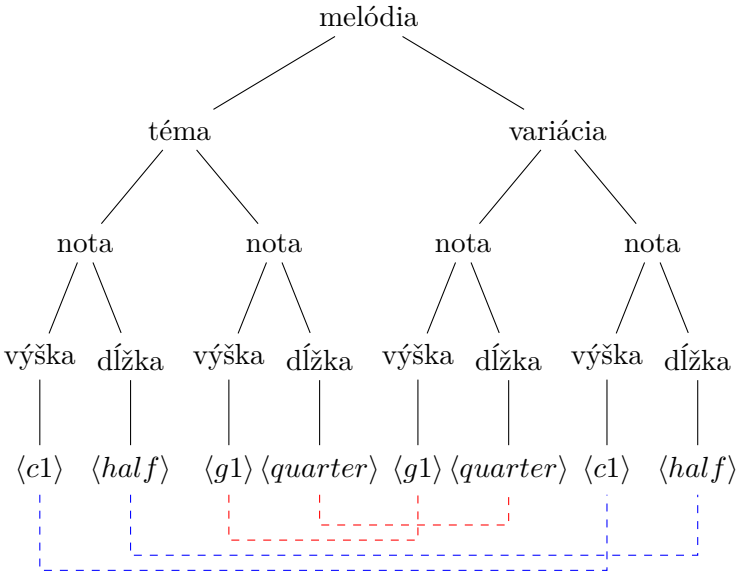
Retrogradácia

Retrogradácia je opakovanie tónov rovnakej výšky a dĺžky, ale v opačnom poradí. Pri reprezentovaní každého tónu jedným terminálom je možné pre retrogradáciu zaviesť bezkontextové pravidlá podobne ako pre generáciu palindrómov. V tejto práci je ale zvolený iný prístup reprezenácie tónov, a preto nie je možné využiť bezkontextové pravidlá.



Obr. 4.9: Retrogradácia motívu.

Na obrázku 4.10 sú zobrazené závislosti jednotlivých terminálov. Prerušovanými čiarami sú naznačené terminály, ktoré je potrebné vygenerovať v jednom derivačnom kroku. Pre výšky tónov môžeme použiť pravidlá navrhnuté pre opakovanie. Pre dĺžky tónov budú definované nové pravidlá, aj keď sa dĺžky tónov nemenia.



Obr. 4.10: Stromová reprezentácia retrogradácie a vzťahov medzi vlastnosťami jednotlivých tónov. Listy reprezentujú melódiu na obrázku 4.9.

V pravidlách potrebujeme vyjadriť poradie tónov, a keďže nový tón nasleduje po terminále pre výšku tónu, definujeme túto skutočnosť v pravidlách pre výšky tónov. Pri variácii retrogradáciou sa z prvého tónu témy stáva posledný tón variácie, za ktorým už ďalšie tóny nenasledujú. To vyjadríme v druhých komponentoch pravidiel pre dĺžku nasledovne.

$$p[2] : (Y) \rightarrow (\langle \text{dĺžka} \rangle),$$

kde $\langle \text{dĺžka} \rangle \in \{whole, half, quarter, eighth\}$. Potom prvé komponenty pravidiel pre dĺžku budú vyzerať nasledovne.

$$p[1] : (Y) \rightarrow (\langle \text{dĺžka} \rangle X X)$$

Podľa tohto vzoru budú pre retrogradáciu zavedené nasledujúce pravidlá.

$$\begin{aligned} &\{(Y,Y) \rightarrow (whole\ X\ X, whole), && (Y,Y) \rightarrow (half\ X\ X, half), \\ &\ (Y,Y) \rightarrow (quarter\ X\ X, quarter), && (Y,Y) \rightarrow (eighth\ X\ X, eighth)\} \subset P \end{aligned}$$

Potom melódiu na obrázku 4.9 je možné pomocou pravidiel

$$\{(S) \to (XX), \quad (X,X) \to (c1\,Y, c1\,Y), \quad (X,X) \to (g1\,Y, g1\,Y),$$
$$(Y,Y) \to (half\ X\ X, half), \quad (Y,Y) \to (quarter\ X\ X, quarter), \quad (X,X) \to (\varepsilon, \varepsilon)\} \subset P$$

zapísať nasledujúcimi derivačnými krokmi.

$$\begin{aligned}
\langle S\rangle &\Rightarrow_G \langle X\rangle\langle X\rangle \\
&\Rightarrow_G \langle c1\rangle\langle Y\rangle\langle c1\rangle\langle Y\rangle \\
&\Rightarrow_G \langle c1\rangle\langle half\rangle\langle X\rangle\langle X\rangle\langle c1\rangle\langle half\rangle \\
&\Rightarrow_G \langle c1\rangle\langle half\rangle\langle g1\rangle\langle Y\rangle\langle g1\rangle\langle Y\rangle\langle c1\rangle\langle half\rangle \\
&\Rightarrow_G \langle c1\rangle\langle half\rangle\langle g1\rangle\langle quarter\rangle\langle X\rangle\langle X\rangle\langle g1\rangle\langle quarter\rangle\langle c1\rangle\langle half\rangle \\
&\Rightarrow_G \langle c1\rangle\langle half\rangle\langle g1\rangle\langle quarter\rangle\langle g1\rangle\langle quarter\rangle\langle c1\rangle\langle half\rangle
\end{aligned}$$

4.2 Algoritmus použitý pre generáciu

Algoritmus použitý pre generáciu variácie je založený na algoritme prezentovanom v článku [8]. Algoritmus ukazuje prístup k spracovaniu gramatiky s rozptýleným kontextom, ktorý využíva iba expanzie na vrchole zásobníka. Používa k tomu takzvaný Delay-List a princíp odloženého vyhodnocovania (anglicky *lazy function evaluation*).

Najprv budú definované pojmy, ktoré sú potrebné pre porozumenie fungovania algoritmu.

Definícia 4.2.1. Majme gramatiku s rozptýleným kontextom $G = (V, T, P, S)$ a pravidlo $p : (A_1, A_2, ..., A_n) \to (x_1, x_2, ..., x_n) \in P$, pričom n je prirodzené číslo. *Indexovanie pravidiel* definujeme ako

$$\begin{aligned}
p[k] &= (A_k) \to (x_k), 1 \leq k \leq n \\
p[k:] &= (A_k, ..., A_n) \to (x_k, ..., x_n), 1 \leq k \leq n \\
p[k] = p[k:] &= \emptyset, k > n.
\end{aligned}$$

Napríklad majme pravidlo $p : (A, B, C) \to (a, b, c)$. Potom

$$\begin{aligned}
p[1] &= (A) \to (a) \\
p[2] &= (B) \to (b) \\
p[2:] &= (B, C) \to (b, c) \\
p[3] = p[3:] &= (C) \to (c) \\
p[4] = p[4:] &= \emptyset.
\end{aligned}$$

Definícia 4.2.2. Rozlišujeme dva typy *generácie*. Generácia symbolu je stupeň derivačného kroku, v ktorom sa symbol nachádza. Generácia pravidla je stupeň derivačného kroku, v ktorom je dané pravidlo použité.

Algoritmus priraďuje symbolom na zásobníku a odloženým pravidlám *generáciu*. Tým, že sa vyhodnocuje iba časť derivačného kroku, je potreba identifikovať časti derivačného kroku, ktoré už sú expandované na zásobníku s odloženou časťou pravidla, podľa ktorého sa daný derivačný krok vykonáva.

Definícia 4.2.3. Majme generáciu g a reťazec $x = x_1x_2...x_n$, kde $x_1, x_2, ..., x_n \in V$. Potom *reversal* značí funkciu, ktorá otočí reťazec x a každému symbolu priradí generáciu g.

$$reversal(x, g) = \langle x_n, g\rangle ... \langle x_2, g\rangle \langle x_1, g\rangle$$

Definícia 4.2.4. Vyhľadávacia štruktúra *Delay-list* reprezentuje kolekciu kľúčov a hodnôt. Ako kľúč používame dvojicu $\langle$ *generácia, neterminál* $\rangle$ a nespracované komponenty pravidla ako hodnotu. *Delay-list*$[N, g]$ určuje odloženú časť pravidla s najnižšou generáciou väčšou než g, ktoré má neterminál N na ľavej strane prvej komponenty.

Príklad 4.2.1. Majme nasledujúci *Delay-list*

$$\langle 2, (X) \to (aX)\rangle$$
$$\langle 3, (X) \to (\varepsilon)\rangle$$

Potom *Delay-list*$[X, 1] = (X) \to (aX)$.

Algoritmus ďalej využíva prediktívnu LL-tabuľku. Ak $G = (N, T, P, S)$ je gramatika s rozptýleným kontextom a P_1 je multimnožina taká, že $P_1 = \{p[1], p \in P\}$, potom G je LL gramatika s rozptýleným kontextom, ak $G_1 = (N, T, P_1, S)$ je LL bezkontextová gramatika. Prediktívna LL-tabuľka je dvojrozmerná vyhľadávacia štruktúra indexovaná neterminálom A z terminálom a. Potom *LL-table[A,a]* obsahuje pravidlo, ktoré musí byť uplatnené ak na vrchole zásobníka je A a aktuálny symbol na vstupnej páske je a.

Pravidlá sú zostavené tak aby ich prvé komponenty obsahovali terminály témy a druhé komponenty terminály variácie. Symboly na zásobníku budú mať priradené typy podľa toho, či sa jedná o neterminál, terminál variácie alebo terminál témy. Tak ako sa pri vkladaní na zásobník priradzuje symbolom generácia, priradí sa im aj tento typ. Keďže terminály témy sú obsiahnuté iba v prvých komponentách pravidiel a tie sa dostanú na zásobník iba pri prvom výbere z LL-tabuľky, na zásobník sa vkladajú ako terminály témy. Keď uplatňujeme pravidlo vybrané z Delay-listu, vkladajú sa na zásobník ako terminály variácie.

Algoritmus využíva zoznam odložených častí pravidiel, tzv. Delay-list, vďaka ktorému prebieha expanzia vždy iba na vrchole zásobníka.

Napríklad na riadku 27 sa zamení vrchol zásobníka za prvú komponentu pravej strany pravidla p, ktoré sme získali z Delay-listu. Potom v Delay-liste nahradíme získané pravidlo tým istým, ale bez prvých komponent oboch strán.

Vstupom algoritmu 1 je LL-tabuľka a reťazec terminálov, ktoré reprezentujú tému. Výstupom je reťazec terminálov, ktorý reprezentuje variáciu. Na začiatku sú inicializované premenné *generácia* (riadok 1) a *variácia* (riadok 2). Na riadku 3 je zásobník inizcializovaný symbolmi $\langle \$, 0\rangle$ $\langle S, 0\rangle$, kde \$ reprezentuje dno zásobníka a S reprezentuje štartovací symbol. Každý symbol na zásobníku má priradené číslo, ktoré reprezentuje jeho generáciu (4.2.2). Algoritmus v cykle `while` prechádza vstupný reťazec a postupne robí expanzie na zásobníku. Podľa toho, aký symbol je na vrchole zásobníka.

Ak je na vrchole zásobníka symbol \$, ktorý značí dno zásobníka, potom ak aktuálny token je tiež symbol \$, ktorý značí koniec vstupnej pásky a Delay-list je prázdny, úspešne prebehli všetky expanzie na zásobníku, výstupom je variácia a algoritmus končí. Inak algoritmus končí chybou.

Ak je na vrchole zásobníka terminál témy, potom ak rovnaký symbol je práve na vstupnej páske, zásobník vykoná operáciu pop a prečíta sa ďalší symbol zo vstupnej pásky. Inak skončí chybou.

Algoritmus 1: Tabuľkou riadené spracovanie gramatiky s rozptýleným kontextom

Input: LL-tabuľka pre gramatiku $G = (N, T, P, S)$; *téma* $\in T^*$
Output: *téma* a *variácia*, ak *téma*,*variácia* $\in L(G)$; inak *chyba*

```
1  generácia := 0;
2  variácia := [];
3  inicializácia zásobníka na ⟨$, 0⟩ ⟨S, 0⟩;
4  inicializácia Delay-listu;
5  while zásobník nie je prázdny do
6      nech ⟨X, g⟩ je vrchol zásobníka a a aktuálny token;
7      if X = $ then
8          if Delay-list je prázdny a súčasne a = $ then
9              ulož_variáciu();
10             pop(⟨X, g⟩);
11         else
12             chyba;
13         end if
14     else if X ∈ T_T then
15         if X = a then
16             pop(⟨X, g⟩);
17             prečítaj ďalší token a zo vstupu;
18         else
19             chyba;
20         end if
21     else if X ∈ T_V then
22         variácia += X;
23         pop(⟨X, g⟩);
24     else if X ∈ N then
25         if Delay-list[X,g] existuje then
26             značˇ Delay-list[X,g] ako ⟨p : (X, X_2, ..., X_n) → (x, x_2, ..., x_n), g'⟩;
27             zameň na zásobníku ⟨X, g⟩ za reversal(x, g');
28             zameň Delay-list[X,g] za ⟨g', p[2 :]⟩;
29         else if p : (X, X_2, ..., X_n) → (x, x_2, ..., x_n) ∈ LL-tabuľka[X, a] then
30             generácia += 1;
31             zameň na zásobníku ⟨X, g⟩ za reversal(x,generácia);
32             vlož ⟨generácia, p[2 :]⟩ do Delay-listu;
33         else
34             chyba;
35         end if
36     else
37         chyba;
38     end if
39 end while
```

Ak je na vrchole zásobníka terminál variácie zapíšeme tento terminál do premennej *variácia*.

Ak je na vrchole zásobníka neterminál, skúsime nájsť odložené komponenty podľa kľúča $\langle X, g \rangle$. Ak také komponenty existujú zameníme vrchol zásobníka za pravú stranu prvej komponenty získaného pravidla a zvyšné komponenty pravidla vrátime do Delay-listu. Ak nenájdeme pravidlo v Delay-liste, pokúsime sa nájsť pravidlo v LL-tabuľke, expandovať zásobník podľa prvej komponenty a zvyšné komponenty vložíme do Delay-listu. Inak chyba.

Kapitola 5

Implementácia

Táto kapitola popisuje implementáciu aplikácie na generovanie hudobných reťazcov podľa vstupného reťazca. Na generáciu využíva algoritmus popísaný v kapitole 4. Kapitola uvádza formát vstupu a výstupu, použitý jazyk a knižnice. V závere kapitoly sú prezentované ukážky vstupov a výstupov.

5.1 Výber jazyka

Ako implementačný jazyk bol zvolený Python3 [11] pre svoju jednoduchosť a dostupnosť knižníc, ktoré ponúkajú nástroje pre analýzu notového zápisu. Použité boli moduly štandardnej knižnice `argparse`, `enum`, `dataclasses` a knižnica `music21` [6].

5.2 Štruktúra výslednej aplikácie

Program je implementovaný ako konzolová aplikácia. Vstupom aj výstupom sú súbory reprezentujúce notový zápis. Vstupný súbor je určený povinným argumentom pri spustení aplikácie.

Podporované formáty pre vstupné súbory sú MusicXML [2] a vlastný formát súboru navrhnutý pre účely tejto práce. Tento formát je popísaný v sekcii 5.3.

Výstup je možné uložiť vo formáte MusicXML a s použitím programu Lilypond [1] aj vo formátoch pdf, png a svg. Typy variácií, ktoré budú vygenerované zvolí užívateľ pomocou prepínačov. Môže zvoliť opakovanie (`-r`), tonálnu sekvenciu nahor (`-u`) a nadol (`-d`), reálnu sekvenciu nahor (`-W`) a nadol (`-w`) predĺženie (`-A`) alebo skrátenie tónov (`-D`) a otočenie melódie (`-R`). Výsledné variácie je možné prehrať (spustenie s prepínačom `--play`).

Prehľad modulov a tried

Program je rozdelený do modulov podľa ich funkcie. Táto podsekcia predstavuje prehľad jednotlivých modulov.

`music_generator.py`

Vstupný bod programu. Implementuje funkciu `main`, ktorá prepája ostatné moduly. V tomto module je implementovaná trieda `ArgParser`, ktorá využíva modul `argparse` jazyka Python pre spracovanie argumentov príkazového riadku.

`scg.py`

Modul obsahuje triedu, ktorá implementuje generáciu variácií a ďalšie pomocné triedy.

- `Rule` je dátová trieda, ktorá definuje ľavú a pravú stranu pravidla.
- `Variation` je dátová trieda, ktorá definuje LL-tabuľku variácie.
- `InputString` je dátová trieda, ktorá definuje vstupnú pásku. Implementuje metódy pre čítanie aktuáneho symbolu, nasledujúceho symbolu a návrat na začiatok.
- `DelayList` implementuje Delay-list ako slovník a metódy pre vloženie, získanie pravidla a zámenu pravidla v Delay-liste.
- `Pushdown` implementuje zásobník a jeho operácie.
- `Symbol` definuje symbol na zásobníku. Symbol má `data`, `generation` a `symbolType` – potrebné pre rozlíšenie, či ide o symbol variácie alebo témy.
- `SymbolType` je výčtového typu, a definuje typ symbolu na zásobníku, čo môže byť `TERMINAL`, `NONTERMINAL` alebo `VARIATION_TERMINAL`.
- `Parser` je hlavnou triedou modulu. Implementuje algoritmus 1 popísaný v kapitole 4. Využíva vyššie spomenuté triedy pre Delay-list a zásobník. Implementuje pomocné metódy, ktoré využíva hlavná metóda `parse()`.

`fileHandler.py`

Implementuje triedu `MusicParser`, ktorá obsahuje metódy na prácu so vstupným súborom, na získanie tokenov z neho a potom na uloženie výslednej variácie do súboru. Modul využíva knižnicu `music21`, ktorá ponúka nástroje na prácu s formátom MusicXML.

`musicGrammar.py`

Definuje vstupnú abecedu, pravidlá a LL-tabuľky definovanej gramatiky pre generáciu variácií. Využíva k tomu triedy modulu `scg.py` – `Rule` a `Variation`. Ďalej implementuje triedu `TokenConverter`, ktorá konvertuje symboly medzi ich reprezentáciou v objektoch knižnice `music21` a ich reprezentáciou v navrhnutých pravidlách. Tiež implementuje metódu na konverziu symbolov zo vstupného súboru na terminály navrhnutej gramatiky.

Reprezentácia gramatiky

Terminály a neterminály sú reprezentované reťazcami (v Pythone dátový typ `str`) tak, ako boli navrhnuté v kapitole 4. Vstupná abeceda je definovaná v zozname `input_alphabet`. Pravidlá navrhnuté v kapitole 4 sú implementované ako objekty triedy `Rule`, ktorá má atribúty pre ľavú a pravú stranu pravidla. Všetky navrhnuté pravidlá sú uložené v zozname `rules`. LL-tabuľky sú implementované formou slovníka ako objekty triedy `Variation`. Kľúčom je dvojica `(nonterminal, terminal)`, kde `nonterminal` bude neterminál na vrchole zásobníka a `terminal` je aktuálny symbol vstupnej pásky. Hodnotou je index do zoznamu `rules` a určuje pravidlo, ktoré sa má aplikovať. Navrhnuté LL-tabuľky sú uložené v slovníku `ll_tables`, kde kľúčom je názov typu variácie a hodnotou je prislúchajúci objekt triedy `Variation`.

Spracovanie vstupu

Spracovanie vstupných argumentov programu zabezpečuje trieda `ArgParser`. Využíva k tomu knižnicu `argparse` jazyka Python. Metódou `parse_args()` spracuje argumenty a pomocou metódy `set_keys_from_program_args()` uloží do atribútu triedy zoznam kľúčov zvolených variácií. Každá variácia má priradený svoj kľúč, pomocou ktorého neskôr zo slovníka `ll_tables` získavame LL-tabuľku danej variácie.

Vstupný súbor (resp. cesta k vstupnému súboru) je zadaný povinným argumentom. Formát vstupného súboru je bližšie popísaný v sekcii 5.3. Spracovanie vstupného súboru zabezpečuje trieda `MusicParser`.

Metóda `from_file(file)` je hlavnou metódou pre spracovanie súboru. Pomocou ďalších metód načíta vstupný súbor a extrahuje z neho taktové značenie a symboly, ktoré reprezentujú tému a uloží ich do príslušných atribútov triedy. Taktové značenie je potrebné pre generáciu výstupného súboru. Extrahované symboly sú uložené ako zoznam `music_tokens` a budú použité ako vstupný reťazec pre `Parser` modulu `scg.py`.

Ak je vstupný súbor vo formáte MusicXML, sú na jeho spracovanie použité objekty knižnice `music21` a metóda `music21.converter.parse()`, ktorá pri úspechu vráti objektovú reprezentáciu melódie. MusicXML aj objekty knižnice `music21` používajú inú konvenciu pre pomenovanie nôt. Na preklad do zvolenej reprezentácie pre terminály je použitá trieda `TokenConverter`.

Generácia pomocou gramatiky s rozptýleným kontextom

Generáciu variácií implementuje trieda `Parser`. Trieda `Parser` je inicializovaná navrhnutou gramatikou, ktorá je definovaná v module `musicGrammar.py`. Potom podľa zadaného vstupného reťazca a zvolených typov variácií generuje reťazce reprezentujúce dané variácie a ukladá ich do atribútu triedy `variations_results`.

Po nastavení atribútov triedy je z `main` volaná metóda `generate_variations()`. Tá v cykle `for` postupne pre každú užívateľom zvolenú variáciu volá metódu `parse(variation)`, kde `variation` definuje LL-tabuľku pre danú variáciu. Metóda `parse()` je hlavnou metódou modulu a implementuje algoritmus navrhnutý v sekcii 4.2. Využíva k tomu vstupnú pásku, zásobník a zoznam odložených pravidiel. Ich implementácia je popísaná v nasledujúcich sekciách.

Metóda `push_reversal_to_pushdown(rhs_first_component, generation, pushing_variation)`

Trieda `Parser` obsahuje pomocnú metódu `push_reversal_to_pushdown`, ktorá implementuje reverzáciu reťazca symbolov a priradenie generácie ako v definícii 4.2.3. Parametrom metódy je pravá strana prvého komponentu pravidla, ktorú je potrebné otočiť a vložiť na zásobník.

Metóda navyše symbolom priraďuje typ pre rozlíšenie terminálov témy a variácie. Pravidlá pre generáciu variácií sú navrhnuté tak, aby ich prvé komponenty obsahovali iba terminály témy a druhé komponenty terminály variácie. Podľa toho, či expanzia zásobníka prebieha pomocou odloženého pravidla alebo pravidla vybraného z LL-tabuľky je možné rozhodnúť, ktoré komponenty pravidla sa budú vkladať na zásobník. Potom parametrom `pushing_variation` je určené, či majú byť symboly vložené ako terminály témy alebo variácie.

Typ symbolu je implementovaný triedou `SymbolType(Enum)`, kde `NONTERMINAL` je neterminál, `TERMINAL` je terminál témy a `VARIATION_TERMINAL` je terminál variácie.

Vstupná páska – trieda `InputString`

Vstupná páska je implementovaná ako zoznam symbolov na vstupnej páske v atribúte `string`. Aktuálny symbol je určený indexom `current_index`, pričom `current_index = 0` značí začiatok pásky. Trieda implementuje metódy:

- `get_current_token()` – Metóda pre získanie aktuálneho symbolu.
- `read_token()` – Metóda prečíta nasledujúci symbol na páske a vráti ho.
- `is_parsed()` – Metóda na zistenie, či je prečítaná celá páska.
- `reset()` – Nastaví čítanie pásky na prvý symbol, teda `current_index = 0`

Zásobník – trieda `Pushdown`

Implementovaný algoritmus pre generáciu využíva jednoduchý zásobník a pracuje iba s jeho vrcholom. Zásobník je implementovaný v triede `Pushdown`. Obsahuje objekty triedy `Symbol`.

Symboly na zásobníku majú generáciu a typ ako bolo navrhnuté v kapitole 4. Tieto informácie sú uložené v objektových atribútoch `symbolType` a `generation`. Generácia je reprezentovaná číslom a typ symbolu je implementovaný triedou `SymbolType`.

Ďalej objektový atribút `data` uchováva informáciu o tom, ktorý konkrétny symbol vstupnej abecedy na zásobníku reprezentuje. Nakoniec, keďže je zásobník implementovaný ako previazaný zoznam, každý objekt `Symbol` má odkaz na nasledujúci symbol v objektovom atribúte `next_symbol`.

Zásobník je inicializovaný symbolmi:

- `$` – pre dno zásobníka
- `S` – štartovací symbol

Oba symboly majú pridelenú generáciu `0`. Symbol na vrchole zásobníka je uchovávaný v atribúte `top` objektu triedy `Pushdown`. Po inicializácii je na vrchole zásobníka symbol `S`.

Trieda reprezentujúca zásobník implementuje metódy pre zásobníkové operácie `push` a `pop`.

Metóda `push(symbolType, data, generation)`

Metóda implementuje vkladanie na vrchol zásobníka. Zadané parametre definujú typ symbolu (`symbolType`), generáciu (`generation`) a symbol (`data`). Podľa zadaných parametrov vytvorí objekt `Symbol`, nastaví nasledujúci symbol a uloží ho do atribútu `top`. Túto metódu využíva metóda `push_reversal_to_pushdown` triedy `Parser` pri expandovaní zásobníka.

Metóda `pop()`

Metóda implementuje odobratie symbolu z vrcholu zásobníka. Metóda `pop` odstráni zo zásobníka symbol, ktorý je na jeho vrchole a vráti hodnotu jeho atribútu `data`. Ak je na vrchole zásobníka terminál variácie, pri vykonaní operácie `pop` je jej návratová hodnota uložená do premennej `variation_result` pre výsledné variácie.

Zoznam odložených pravidiel – trieda `DelayList`

Trieda implementuje Delay-List definovaný v sekcii 4.2. Samotný Delay-List je reprezentovaný ako slovník v atribúte `delay_list_dictionary`. V štruktúre sa vyhľadáva podľa generácie pravidla a podľa neterminálu, ktorý je v prvej komponente ľavej strany pravidla. Preto je kľúčom dvojica `(generation, nonterminal)` a hodnotou je pravidlo (resp. odložená časť pravidla).

Metóda `get_delayed_rule(nonterminal, generation_of_rule)`

Metóda implementuje vyhľadávanie v Delay-Liste z definície 4.2.4. V cykle `for` iteruje cez kľúče Delay-Listu a hľadá kľúč `r_key`, ktorý vyhovuje podmienke:

```
r_key[1] == nonterminal and r_key[0] > generation_of_rule and
(return_min_generation is None or r_key[0] < return_min_generation)
```

Teda hľadáme kľúč, ktorého neterminál sa zhoduje s vyhľadávaným neterminálom, súčasne generácia v kľúči je väčšia než generácia, podľa ktorej vyhľadávame a súčasne musí byť táto generácia najmenšia z tých, ktoré vyhovujú predchádzajúcej podmienke. Nakoniec metóda vrátí nájdené pravidlo a jeho generáciu ako v návrhu algoritmu 1 na riadku 26.

Metóda `push_rule(rule, generation)`

Metóda implementuje vloženie pravidla do Delay-Listu. Podľa pravidla `rule` a generácie `generation` vytvorí kľúč a vloží pravidlo `rule` bez prvých komponentov do Delay-Listu.

Metóda `replace_delayed_rule(nonterminal, rule_generation, rule)`

Metóda vymaže z Delay-Listu pravidlo podľa neterminálu `nonterminal` a generácie `rule_generation` a pomocou metódy `push_rule()` ho nahradí.

Výstup aplikácie

Po vygenerovaní je možné variácie uložiť alebo zobraziť. To zabezpečuje metóda `to_file` triedy `MusicParser`. Základným chovaním je výpis variácií do konzoly vo formáte TinyTheme (popísaný v sekcii 5.3). Pri spustení s prepínačom `--save` bude namiesto výpisu výstup v tomto formáte uložený do súboru s príponou `.txt`. Názov a umiestnenie výstupného súboru je určené podľa vstupného súboru. Umiestnenie je rovnaké a k názvu sa pridá reťazec `_variations` a príslušná prípona súboru. Transformáciu vygenerovaných variácií na reprezentáciu vo formáte TinyTheme zabezpečuje metóda `generate_tinytheme_output()`.

Pre uloženie v iných formátoch je potrebné symboly variácie transformovať na objekty knižnice `music21`. Použité triedy sú:

`music21.stream.Part` – reprezentuje výslednú skladbu
`music21.stream.Measure` – reprezentuje jeden takt
`music21.stream.Note` a `music21.stream.Rest` – reprezentujú notu a pomlčku

Metóda `generate_music21_part` vytvorí objekt `Part`, ktorý postupne napĺňa taktmi a do nich vhodne rozdeľuje noty a pomlčky.

Pri variáciách so zmenou dĺžky sa môže stať, že dĺžka noty presahuje do druhého taktu. Pri reprezentácií TinyTheme to nevadí, pretože melódiu do taktov nedelí. Pri transformácií do vyššie uvedených objektov sa o to stará metóda `generate_music21_part` a jej pomocné

metódy, ktoré určia aká dĺžka sa ešte do taktu vojde a zvyšok dĺžky presunú do nasledujúceho taktu. Tieto dve noty sú navyše prepojené ligatúrou (oblúčik spájajúci noty rovnakej výšky). Výsledkom je jeden tón ktorý má trvanie tónov, ktoré ligatúra spája.

Výsledné variácie je potom možné uložiť vo formáte MusicXML (spustenie pomocou `--save musicxml`) a s použitím programu Lilypond aj vo formátoch pdf, svg a png. Knižnica `music21` umožňuje výsledné variácie prehrať, spustením programu s prepínačom `--play`.

5.3 Formát vstupného súboru

Program podporuje vstup vo formáte MusicXML. Formát MusicXML je podporovaný mnohými aplikáciami a predstavuje vhodný prostriedok na zdieľanie aj rozsiahlych hudobných diel v textovom formáte. Umožňuje zápis komplikovaných hudobných štruktúr, čoho výsledkom je aj zložitejšia syntax. MusicXML je preto vhodný najmä na import a export v aplikáciách pre notový zápis.

Tento formát súboru sa však ukázal ako nevhodný, kvôli svojej komplexnosti. Preto, s cieľom umožniť čo najjednoduchší zápis melódie, bol pre aplikáciu navrhnutý textový formát založený na formáte TinyNotation knižnice music21. Zápis TinyNotation bol vyvinutý s rovnakým zámerom, aby užívateľ mohol jednoducho a rýchlo textovo zapísať melódiu bez použitia iných notových editorov a bez nutnosti pamätať si komplikovanú syntax. Pre rozlíšenie budeme tento textový formát nazývať TinyTheme.

TinyTheme – syntax

Formát vyžaduje, aby na prvom riadku súboru boli iba hlavička `TinyTheme:` a taktové značenie. V hlavičke sa nerozlišujú veľké a malé písmená.

Správne	**Nesprávne**
`TinyTheme: 3/4`	`TinyTheme : 3/4`
`tInYthEmE:4/4`	`tInYthEmE:` `4/4`
`tinytheme: 4/4`	`tinytheme: 4/4 c1_1`

Tabuľka 5.1: Ukážky správnych a nesprávnych zápisov hlavičky a taktového značenia vo formáte TinyTheme.

Taktové značenie zapisujeme ako zlomok bez medzier. Napríklad štvorštvrťový takt zapíšeme ako `4/4`. Na ďalších riadkoch súboru nasledujú symboly reprezentujúce jednotlivé noty. Tieto symboly môžu byť oddelené ľubovoľným počtom bielych znakov. Symboly reprezentujúce noty budú formátu ⟨výška_tónu⟩_⟨dĺžka_tónu⟩. Časť symbolu, ktorá reprezentuje výšku sa bude značiť malým písmenom a číslom. Časť, ktorá reprezentuje dĺžku

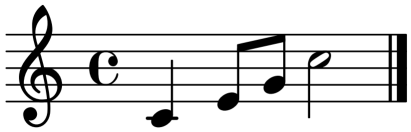
tónu sa značí číslom podobne ako v taktovom značení. V tabuľke 5.2 sú ukázané noty ako sa reprezentujú.

	c^1	d^1	e^1	f^1	g^1	a^1	h^1	c^2	d^2	e^2	f^2	g^2	a^2	h^2	c^3
celá	`c1_1`	`d1_1`	`e1_1`	`f1_1`	`g1_1`	`a1_1`	`h1_1`	`c2_1`	`d2_1`	`e2_1`	`f2_1`	`g2_1`	`a2_1`	`h2_1`	`c3_1`
polová	`c1_2`	`d1_2`	`e1_2`	`f1_2`	`g1_2`	`a1_2`	`h1_2`	`c2_2`	`d2_2`	`e2_2`	`f2_2`	`g2_2`	`a2_2`	`h2_2`	`c3_2`
štvrťová	`c1_4`	`d1_4`	`e1_4`	`f1_4`	`g1_4`	`a1_4`	`h1_4`	`c2_4`	`d2_4`	`e2_4`	`f2_4`	`g2_4`	`a2_4`	`h2_4`	`c3_4`
osminová	`c1_8`	`d1_8`	`e1_8`	`f1_8`	`g1_8`	`a1_8`	`h1_8`	`c2_8`	`d2_8`	`e2_8`	`f2_8`	`g2_8`	`a2_8`	`h2_8`	`c3_8`

Tabuľka 5.2: Formát pre reprezentáciu nôt v TinyTheme.

Pomlčky sú reprezentované podobne, ale namiesto značenia pre výšku tónu používajú znak `r`. Teda pre celú, polovú, štvrťovú a osminovú pomlčku sú použité značenia `r_1`, `r_2`, `r_4` a `r_8`.

```
1 TinyTheme:4/4
2
3 c1_4 e1_8 g1_8 c2_2
```



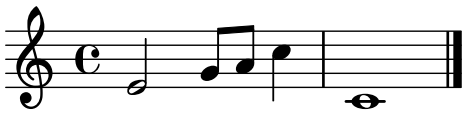
Obr. 5.1: Príklad vstupného súboru vo formáte TinyTheme a jeho reprezentácia v notovom zápise.

Výstup je možné tiež uložiť vo formáte TinyTheme. To umožňuje pri generácii variácii, ktoré ostávajú v tónine c-dur (všetky navrhnuté variácie okrem reálnych sekvencií – prepínače `-w` a `-W`), výsledok opäť variovať.

5.4 Vyhodnotenie – ukážky výstupov

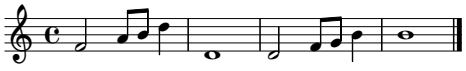
V tejto sekcii budú prezentované ukážky vstupov a výstupov programu. Ukážky budú prezentované vo formáte TinyTheme a ich reprezentácia v notovom zápise.

```
1 tinyTheme: 4/4
2
3 e1_2 g1_8 a1_8 c2_4 c1_1
```



Obr. 5.2: Krátky motív reprezentovaný vo formáte TinyTheme a jeho reprezentácia v notovom zápise.

```
1 tinyTheme: 4/4
2
3 f1_2 a1_8 h1_8 d2_4 d1_1
4
5 d1_2 f1_8 g1_8 h1_4 h1_1
```



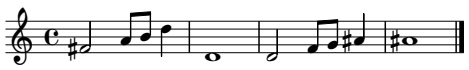
Obr. 5.3: Tonálna sekvencia nahor a nadol.

Na obrázku 5.3 je prezentovaná tonálna sekvencia motívu z obrázka 5.2 nahor (prvý a druhý takt) a nadol (tretí a štvrtý takt). Pri sekvencii nadol pri neobmedzenom rozsahu

by mala byť posledná nota *malé h*. Vygenerovaná je však nota h^1 tak, ako bolo navrhnuté pre pravidlá, ktoré by inak dostávali melódiu mimo zvolený rozsah.

Na obrázku 5.4 je prezentovaná reálna sekvencia motívu z obrázka 5.2. Tóny sú oproti tónom melódie posunuté vždy o celý tón okrem posledného tónu sekvencie smerom nadol. Pri ňom nastáva podobná situácia ako v predchádzajúcom príklade. Aplikované bolo pravidlo, ktoré je navrhnuté tak, aby generovalo tón, ktorý patrí do zvoleného rozsahu.

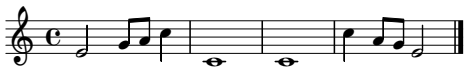
```
1 tinyTheme: 4/4
2
3 fis1_2 a1_8 h1_8 d2_4 d1_1
4
5 d1_2 f1_8 g1_8 ais1_4 ais1_1
```



Obr. 5.4: Reálna sekvencia nahor a nadol.

Na obrázku 5.5 je prezentovaná ukážka opakovania motívu z obrázku 5.2 a za ňou retrogradácia motívu. V notovej reprezentácií aj v reprezentácii TinyTheme je vidieť, že pri variácií opakovaním sú noty motívu (obr. 5.2) rovnaké. Ďalej je vidieť, že noty retrogradácie tvoria zrkadlovú štruktúru. V reprezentácii TinyTheme riadok 3 reprezentuje variáciu opakovaním teda aj tóny pôvodného motívu. Na riadku 5 sú reprezentované tóny variácie retrogradáciou. Pri čítaní symbolov riadku 3 odzadu získame rovnakú postupnosť symbolov ako na riadku 5.

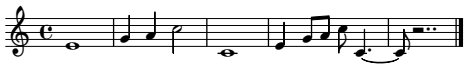
```
1 tinyTheme: 4/4
2
3 e1_2 g1_8 a1_8 c2_4 c1_1
4
5 c1_1 c2_4 a1_8 g1_8 e1_2
```



Obr. 5.5: Opakovanie motívu a retrogradácia.

Na obrázku 5.6 je zobrazená augmentácia a diminúcia motívu z obrázku 5.2. Pôvodný motív má dĺžku dvoch štvorštvrťových taktov. Pri augmentácii teda očakávame variáciu o dĺžke štyroch taktov a pri diminúcii variáciu s dĺžkou jedného taktu. V pravidlách pre augmentáciu je pre celú notu zvolené pravidlo pre opakovanie a podobne pri diminúcii pre notu osminovú. V motíve je posledná nota celá. Tá si pri augmentácii zachováva svoju dĺžku (obr. 5.6 – tretí takt). V motíve sú tiež dve osminové noty, ktoré si zachovávajú svoju dĺžku pri diminúcii. Posledný tón pri diminúcií vyšiel s presahom do nasledujúceho taktu. To je vyriešené rozdelením na dva tóny, ktoré majú v súčte požadovanú dĺžku tak ako bolo popísané v sekcii 5.2.

```
1 tinyTheme: 4/4
2
3 e1_1 g1_4 a1_4 c2_2 c1_1
4
5 e1_4 g1_8 a1_8 c2_8 c1_2
```



Obr. 5.6: Augmentácia a diminúcia.

Kapitola 6

Záver

Cieľom tejto práce bolo vytvoriť aplikáciu, ktorá s využitím formálnych modelov bude generovať variácie zadaného motívu. Pre splnenie cieľa boli študované modely podľa Chomského hierarchie, ktoré tvoria základ pochopenia teórie formálnych modelov. Hlavným typom gramatík, na ktoré sa práca zameriava sú gramatiky s rozptýleným kontextom. Tie svojimi vlastnosťami vyhovujú na generáciu hudobných štruktúr ako je téma a jej variácia.

Pre generáciu štruktúry pre motív a jeho variáciu boli navrhnuté gramatiky s rozptýleným kontextom. Gramatiky boli navrhnuté pre variáciu opakovaním, pre tonálne a reálne sekvencie nahor a nadol, pre skrátenie a predĺženie tónov a pre retrogradáciu.

V práci sú tieto modely implementované pre generovanie variácie podľa témy. Algoritmus na generáciu je založený na algoritme pre tabuľkou riadené spracovanie gramatiky s rozptýleným kontextom. Využíva LL-tabuľky pridelené každej variácii, zoznam odložených pravidiel a zásobník. Vďaka zoznamu odložených pravidiel je možné pracovať vždy iba s vrcholom zásobníka.

Výsledná aplikácia podporuje vstupné formáty MusicXML a vlastný vstupný formát navrhnutý pre účely tejto práce. V týchto formátoch je podporované aj uloženie výstupu aplikácie. Ďalej aplikácia ponúka možnosť výsledné variácie prehrať. To umožňuje vygenerované variácie použiť ako vstup pri opätovnom spustení a tak tvoriť ďalšie variácie.

Program bol po implementácii otestovaný pomocou rôznych vstupných súborov. Testovanie ukázalo, že navrhnuté pravidlá a generácia variácií fungujú správne.

Potenciál na zlepšenie vidím napríklad v automatizovaní procesu, kedy je variáciu možné opäť variovať. Program bol navrhnutý tak, aby bolo možné navrhnuté pravidlá nahradiť vlastnými. Samotné pravidlá by bolo možné rozšíriť o podporu väčšieho rozsahu tónov alebo iné dĺžky tónov. Teoreticky je možné pridať modul, ktorý by obsahoval pravidlá pre iné variácie než boli navrhnuté a implementované v tejto práci.

Využitie aplikácie vidím napríklad v prostrediach, kde hudba nie je stredobodom, ale tvorí pozadie. Takéto aplikácie môžu zjednodušovať a automatizovať tvorbu veľkého množstva generickej hudby a zároveň vnášať do nej variácie aby nebola príliš monotónna.

Literatúra

[1] *LilyPond – Music notation for everyone* [online]. 2023 [cit. 2023-07-15]. Dostupné z: https://lilypond.org/.

[2] *MusicXML for Exchanging Digital Sheet Music* [online]. 2023 [cit. 2023-07-15]. Dostupné z: https://www.musicxml.com/.

[3] Chomsky, N. Three models for the description of language. *IRE Transactions on Information Theory*. 1. vyd. 1956, zv. 2, č. 3, s. 113–124. DOI: 10.1109/TIT.1956.1056813.

[4] Chomsky, N. Formal Properties of Grammars. In: Luce, R. D., Bush, R. R. a Galanter, E., ed. *Handbook of Mathematical Psychology*. New York: John Wiley & Sons, 1963, sv. 2.

[5] Cmíral, A. *Základní pojmy hudební*. 7. vyd. Praha: Státní hudební vydavatelství, 1965.

[6] Cuthbert, M. S. A. *Music21* [online]. [cit. 2023-07-15]. Dostupné z: https://pypi.org/project/music21/8.1.0/.

[7] Greibach, S. a Hopcroft, J. Scattered context grammars. *Journal of Computer and System Sciences*. 1969, zv. 3, č. 3, s. 233–247. DOI: https://doi.org/10.1016/S0022-0000(69)80015-2. ISSN 0022-0000. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0022000069800152.

[8] Jirák, O. Table-Driven Parsing of Scattered Context Grammar. In: *Proceedings of the 16th Conference Student EEICT 2010 Volume 5*. Faculty of Information Technology BUT, 2010, s. 171–175. ISBN 978-80-214-4080-7. Dostupné z: https://www.researchgate.net/publication/228952574_Table-Driven_Parsing_of_Scattered_Context_Grammar.

[9] Meduna, A. *Automata and Languages: Theory and Applications*. 1. vyd. London: Springer, 2000. ISBN 978-1-85233-074-3.

[10] 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.

[11] Python Software Foundation. *The Python Language Reference* [online]. [cit. 2023-07-15]. Dostupné z: https://docs.python.org/3/reference/index.html.

[12] Rozenberg, G. a Salomaa, A., ed. *Handbook of Formal Languages, Volume 1: Word, Language, Grammar*. 1. vyd. Springer, 1997. ISBN 978-3-642-63863-3.

[13] Stein, L. *Structure & style: The study and analysis of Musical Forms.* Expanded. Miami, FL: Summy-Birchard Music, 1979. ISBN 0-87487-164-6.

[14] Wikimedia Commons. *File:Hudobne nastroje.svg* [online]. 27. decembra 2008 [cit. 2023-04-12]. Dostupné z: https://commons.wikimedia.org/wiki/File:Hudobne_nastroje.svg.

[15] Zenkl, L. *ABC hudebních forem.* 3. vyd. Praha: EDITIO PRAGA, 1999. ISBN 80-7058-472-6.

Príloha A

Obsah priloženého pamäťového média

src/ – priečinok so zdrojovými súbormi aplikácie

lib/ – priečinok s použitými knižnicami

thesis/ – priečinok obsahujúci zdrojový kód tejto technickej správy

xstrak38.pdf – PDF obsahujúce túto technickú správu

README.md – obsahuje návod k aplikácii