VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

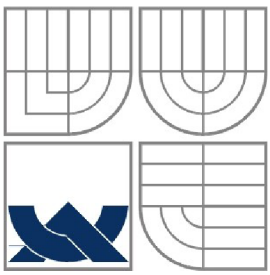
AUTOMATICKÁ TVORBA VARHANNÍ PŘEDEHRY K CÍRKEVNÍM PÍSNÍM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

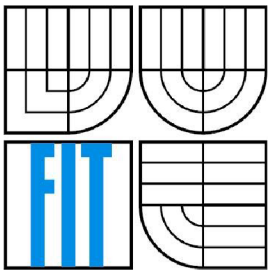
AUTOR PRÁCE
AUTHOR

BC. ONDŘEJ MAŇÁK

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÁ TVORBA VARHANNÍ PŘEDEHRY K CÍRKEVNÍM PÍSNÍM

AUTOMATIC CREATION OF ORGAN OVERTURES FOR CHURCH SONGS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. ONDŘEJ MAŇÁK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MICHAL FAPŠO

BRNO 2010

Abstrakt

Tato diplomová práce se zabývá problematikou automatické tvorby varhanních předeher k církevním písním z teoretického i praktického hlediska. Varhanní předehra je krátký úvod k liturgickému zpěvu. Vzhledem k tomu, že ji lze popsat pomocí konečné množiny pravidel, nabízí se k její tvorbě využít principů úloh s omezujícími podmínkami. Efektivním prostředkem pro vývoj systému potom může být programovací jazyk C++ a knihovna Gecode.

Abstract

The focus of this master's thesis is an automatic creation of organ overtures for church songs from both theoretical and practical points of view. Organ overture is a short introduction to a church song. According to the fact that it can be described by a finite set of rules, it is possible to use techniques for solving Constraint Satisfaction Problems. An effective instrument to develop such system can be C++ programming language and Gecode library.

Klíčová slova

úloha s omezujícími podmínkami, harmonizace melodie, homofonie, polyfonie, Gecode, LilyPond

Keywords

constraint satisfaction problem, melody harmonization, homophony, polyphony, Gecode, LilyPond

Citace

Ondřej Maňák: Automatická tvorba varhanních předeher k církevním písním, diplomová práce, Brno, FIT VUT v Brně, 2010

Automatická tvorba varhanní předehry k církevním písním

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Michala Fapša.
Další informace mi poskytl Mario Buzzi, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Bc. Ondřej Maňák
25. 5. 2010

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu, Ing. Michalu Fapšovi, za pomoc a podporu, kterou mi poskytoval během celého období tvorby mé diplomové práce. Dále bych rád poděkoval Mariovi Buzzi, Ph.D., který mi poskytl konzultace v oblasti hudební problematiky.

Obsah

1 Úvod

Tato práce je textovou částí mého diplomového projektu. Již z tématu vyplývá, že se nejedná o problematiku, na kterou lze nahlížet čistě z hlediska informačních technologií. Je třeba brát v úvahu i hudební teorii, ze které vychází algoritmy použité v programu. Neobešel jsem se tedy bez toho, abych používal termíny, které jsou spojeny s hudební teorií. Tyto termíny jsou stručně vysvětleny ve slovníčku pojmů, který je jednou z příloh tohoto textu. Pro lepší čitelnost textu jsou pojmy, které jsem použil poprvé, vysvětleny rovněž v poznámce pod čarou.

1.1 Motivace

Téma práce jsem si zvolil hned z několika důvodů. Prvním z nich je snaha nějakým způsobem navázat na moji bakalářskou práci Harmonizace melodie [1]. Podstatou této práce byla tvorba systému, který vytváří harmonický doprovod k dané písni. Algoritmus, který program využívá při harmonizaci, odpovídá postupu skladatele. K jednotlivým tónům melodie přidá vždy některý vhodný akord, který náleží odpovídající stupnici. Při tvorbě tohoto programu jsem však narazil na problém, jak do algoritmu programu zahrnout možnost splnění velkého množství pravidel, které mohou ovlivňovat proces harmonizace. Výsledkem je aplikace, která dokáže vytvářet jednoduchou harmonizaci k dané písni. Má však svá omezení. Téma tohoto diplomového projektu zahrnuje i použití harmonizéru. K jeho tvorbě jsem se však rozhodl přistupovat jinou cestou, a to přes úlohy s omezujícími podmínkami.

Systém pro harmonizaci melodie jsem však již jednou tvořil, a proto jsem chtěl své téma nějakým způsobem rozšířit. Jelikož jsem praktikujícím varhaníkem Českobratrské církve evangelické, hraji při nedělních bohoslužbách na varhany k doprovodu společného zpěvu. Každou píseň je třeba vždy uvést krátkou předehrou, kterou není úplně snadné vymyslet. Napadlo mě tedy, že by mohlo být přínosem, kdybych vytvořil aplikaci, která by byla schopna komponovat takové varhanní předehry.

1.2 Význam systémů automatické kompozice

Automatická tvorba hudby je problematika, kterou se v současné době zabývá mnoho projektů, a to jak v akademické, tak i v komerční sféře. Vyvstává tedy otázka, zda bude v budoucnosti třeba hudebních skladatelů, když máme počítače, které se snažíme naučit tvořit hudbu. Nebereme tedy hudebním skladatelům jejich denní chléb, když se snažíme vytvořit něco, co by bylo schopno nahradit jejich práci? To je dosti závažná filozofická otázka.

Myslím si však, že význam automatické kompozice tkví v něčem jiném, než je snaha o nahrazení člověka v procesu tvorby hudby počítačem. Systémy pro automatickou kompozici mají potenciál být dobrými pomocníky hudebního skladatele. Mohou najít uplatnění nejen při samotné tvorbě hudby, ale i při výuce kompozice. Lidský faktor je však stále potřebný pro vývoj nových hudebních stylů i tvorbu jednotlivých skladeb. Zde je třeba mít nějakou myšlenku a tu potom dále rozvíjet. To však zatím počítače neumí.

1.3 Cíle práce

Cílem mé práce tedy bylo vytvořit pomocníka pro varhaníka. Tento pomocník by měl být schopen vytvořit několik varhanních předeher k dané písni. Je potom na varhaníkovi, kterou z nich si vybere pro praktické použití. Druhým způsobem, jak daného pomocníka využít, by potom mohla být možnost experimentovat a vytvořit tak větší množství předeher k dané písni. Uživatel se potom může na těchto příkladech učit, jak ovlivní aplikace různých pravidel výstup systému. Tento přístup předpokládá, že systému lze nastavit na vstupu jisté parametry, které reprezentují některá pravidla pro tvorbu hudby.

Aby tato práce byla přínosem pro varhaníky, musí být program, který je jejím výstupem, prakticky použitelný pro běžného uživatele. Je tedy třeba udělat funkční program, který neklade vysoké nároky na uživatele. Jde totiž o to, aby bylo možno se jednoduše a rychle naučit s programem pracovat. Jinou možností, jak moji práci využít je potom další výzkum v oblasti automatické tvorby hudby. V rámci svého projektu jsem se snažil naplnit oba tyto rozměry.

1.4 Obsah jednotlivých kapitol

Pro snazší orientaci v textu jsem se rozhodl v následujících odstavcích stručně nastínit, čím se zabývají jednotlivé kapitoly této práce.

Kapitola 2 – Pohled hudební teorie se zabývá problematikou tvorby varhanních předeher z pohledu hudební teorie. Jsou zde vysvětleny základní hudební pojmy, které souvisí s touto prací. Dalším předmětem mého zájmu byly principy tvorby varhanní předehry. Těmito principy jsou homofonní a polyfonní přístup. Od těchto přístupů se potom odvíjí jednotlivé styly varhanních předeher. Vše je doplněno názornými příklady v podobě notových ukázek.

Kapitola 3 – Pohled informatiky se zabývá problematikou tvorby varhanních předeher z pohledu informatiky. Nejprve jsou zde představeny různé přístupy k automatické kompozici hudby, které byly v minulosti využity. Dále jsou zde ukázány způsoby reprezentace not pomocí počítače. Následuje rozsáhlejší část, která vysvětluje problematiku úloh s podmínkami. Je tomu tak proto, že principy řešení těchto úloh jsem se rozhodl využít v harmonizace melodie předehry, což je jedna ze stěžejních úloh programové části mé práce. V části, která se zabývá omezujícími podmínkami, je mimo jiné také představena knihovna Gecode pro zpracování těchto úloh a systém Strasheela pro automatickou tvorbu hudby pomocí pravidel, který mi byl velkou inspirací. Dále jsem se zmínil o systémech pro sazbu not, z nichž jsem si vybral LilyPond, a to díky tomu, že umožňuje práci s textovým vstupem.

Kapitola 4 – Návrh systému ukazuje původní návrh celého systému. Je zde řečeno, jaké prostředky byly pro jeho tvorbu použity a vysvětleno, proč jsem se rozhodl využít právě těchto prostředků. Dále jsem se pokusil specifikovat jednotlivé požadavky, které by měl systém splňovat. Nejdůležitějším z těchto požadavků je možnost ovlivnit výstup programu uživatelem prostřednictvím editace konfiguračního souboru. Dále je zde navrženo, jak lze funkcionalitu program rozdělit na jednotlivé části. Funkčnost těchto částí je potom dále specifikována.

Kapitola 5 – Implementace řeší implementaci aplikace, která byla časově nejnáročnější částí mé práce. Je zde vidět, že během implementace se mi úspěšně dařilo držet se původního návrhu celého systému. Je zde nejprve stručně představena funkcionalita jednotlivých modulů, na které je

aplikace rozdělena. K závěru této kapitoly jsem připojil několik málo ukázek výstupu systému v podobě notového zápisu.

Součástí každého většího projektu by měla být vize jeho budoucího vývoje. Protože si myslím, že se jedná o velmi podstatnou záležitost, rozhodl jsem se věnovat jí celou Kapitolu 6 – Budoucí vývoj projektu. Nejprve je navrženo několik možností, jak by mohlo dojít ke zdokonalení systému. Dále je zde zmíněn fakt, že se nám (studentům naší fakulty) podařilo navázat spolupráci s Hudební fakultou Janáčkovy akademie múzických umění. Myslím si, že pro budoucí vývoj projektu bude mít tato spolupráce velký význam.

Kapitola 7 – Závěr je věnována zhodnocení dosažených výsledků mého projektu.

2 Pohled hudební teorie

Při řešení problematiky automatické tvorby varhanních předeher je třeba zohlednit dva přístupy. Prvním z nich je pohled informatiky, kterým jsem se zabýval od kapitoly 3 až do konce této práce. Druhým neméně důležitým přístupem je pohled hudební teorie, kterým jsem se zabýval v této kapitole mé práce. Hudební teorie je totiž věda, která skrývá základní principy, ze kterých je třeba při tvorbě varhanních předeher vycházet i při automatizované tvorbě hudby. Jedná se zejména o práci s hudebním tématem[1], harmonizaci[2] a znalost hudebních forem[3], které mohou být při tvorbě varhanních předeher používány.

Při tvoření varhanních předeher lze potom vycházet ze dvou základních principů. Prvním principem je aplikace homofonie, druhou možností je využití polyfonie. V praxi se však tyto dva přístupy často prolínají a není mezi nimi přesně stanovená hranice. Na základě těchto dvou principů lze tvořit různé styly varhanních předeher, jejichž množství je teoreticky neomezené. V rámci této kapitoly jsem se rozhodl představit některé z těchto stylů. Nejprve je však třeba vymezit si pojem varhanní předehra a představit varhany, jakožto královský nástroj, kterému jsou předehry k církevním písním většinou určeny.

2.1 Varhanní předehra

Vzhledem k tomu, že se tato práce zabývá tvorbou varhanních předeher, což je pojem velmi široký a dosti nejasný, rozhodl jsem se nejprve objasnit, co to vlastně varhanní předehra je. Pojmem varhanní předehra lze označovat ledacos od několika-taktových intonací k církevním písním až po komplikované chorální předehry Johana Sebastiana Bacha, které jsou jedním z vrcholů světové varhanní tvorby. Tyto Bachovy skladby jsou však pro svou náročnost z praktického hlediska jako předehra k liturgickému zpěvu jen stěží použitelné. Mají však své pevné místo na varhanních koncertech.

Význam předehry spočívá v uvedení společného zpěvu při bohoslužbách či mších. Z tohoto uvedení musí být zřejmé, jaká píseň se bude zpívat, v jaké bude tónině a v jakém bude tempu. Jakýmsi nepovinným bonusem je potom navození takové atmosféry, která způsobí, že lidé zpívají s nadšením, chutí a zápalem.

Jako varhanní předehra nám tedy může klidně postačit krátká intonace k dané písni. Takový význam má pojem varhanní předehra v rámci celé této práce. Rozsah předehry není pevně daný, ale většinou se pohybuje v rozmezí od pěti do deseti taktů.

Varhanní předehra může být předem pečlivě složena a zapsána do not i pro další použití jinými varhaníky. To je však dosti pracné a časově náročné. Jiným přístupem, který se často využívá, je tvorba improvizované předehry. Nejčastěji však dochází ke kombinaci těchto dvou přístupů, kdy vznikají jednoduché předem připravené varhanní improvizace.

---

1 Hudební téma = delší závažná myšlenka, která slouží jako kompoziční základ skladby [2].

2 Harmonizace = proces vytváření doprovodných hlasů na základě akordického hudebního myšlení [3].

3 Hudební forma = celkové rozvržení skladby a uspořádání jejích jednotlivých částí včetně hudebních myšlenek [2].

2.2 Varhany

Stejně jako by hudební skladatel měl co nejlépe znát nástroj, pro který píše svou skladbu, měl by i varhaník vědět co nejvíce o varhanách, na kterých doprovází při bohoslužbách či mších. Proto jsem se rozhodl, že se s pomocí knihy Nauka o varhanách, pokusím stručně vysvětlit, co to varhany jsou. Autorem knihy je Vratislav Bělský, který byl kantorem hry na varhany na konzervatoři v Brně. Tento varhaník se rovněž zabýval konstrukcí varhan, kterou rovněž vyučoval v rámci své pedagogické praxe na konzervatoři.

Varhany jsou klávesovým dechovým nástrojem, jehož píšťaly rozeznívá stlačený vzduch. Na rozdíl od ostatních dechových nástrojů musí však být ve varhanách pro každý tón zvláštní píšťala. Spojení několika různě znějících píšťal se zdrojem stálého zvuku bylo známo již ve starověku. Dlouhodobý vývoj ve středověku a renesanci rozšířil technický a zvukový základ nástroje. Vrcholu dosáhly varhany nesporně v období baroka, kdy bylo logicky utříděno zvukové bohatství nástroje. Typickou vlastností varhanního zvuku je jeho neměnnost. Znějící píšťala si zachovává trvale svoji barvu. Změna zvuku je možná pouze změnou rejstříků[4] nebo manuálů[5]. Změnou zvukové barvy dochází samozřejmě také ke změně síly zvuku. Určité dynamické ovlivnění umožňují žaluzie[6] [5].

Zajímavostí je rozmanitost podob jednotlivých varhan, která je u jiných nástrojů v takové míře nevídaná. Můžeme se setkat s malými přenosnými a stejně tak i s monumentálními nástroji, které obsahují tisíce píšťal různých velikostí. Typické je, že na každý nástroj nelze zahrát jakoukoli varhanní literaturu. Některé skladby vyžadují mohutnost velkých nástrojů katedrál, jiné zní lépe na malých nástrojích venkovských kostelíků. Většina nástrojů se však nachází v kostelích a je určena primárně k doprovodu liturgického zpěvu. Předehra k písni a sborový zpěv jsou tedy společným jmenovatelem většiny varhan.

2.3 Homofonní přístup

Homofonie je hudební sloh, v němž bývá jediná melodie hlavní (obvykle se jedná o vrchní hlas) a ostatní hlasy jsou jí podřazeny. Podřazené hlasy nazýváme doprovodem. Skladba se vyvíjí na základě akordickém a uplatňuje se zde vertikální hudební myšlení [3]. Obecný postup při tvorbě předehry s využitím homofonního přístupu je následující:

- tvorba tématu předehry,
- harmonizace tohoto tématu,
- aplikace daného stylu předehry.

Podrobnějšímu popisu celé problematiky se věnuje tato kapitola. Pro názornost jsem se rozhodl vše demonstrovat na příkladu písně Narodil se Kristus Pán (viz Obr. 2-1). Pro tuto píseň jsem se rozhodl proto, že ji zná praktický každý, tudíž je na jejím příkladu dobře vidět, jak byla zpracována.

---

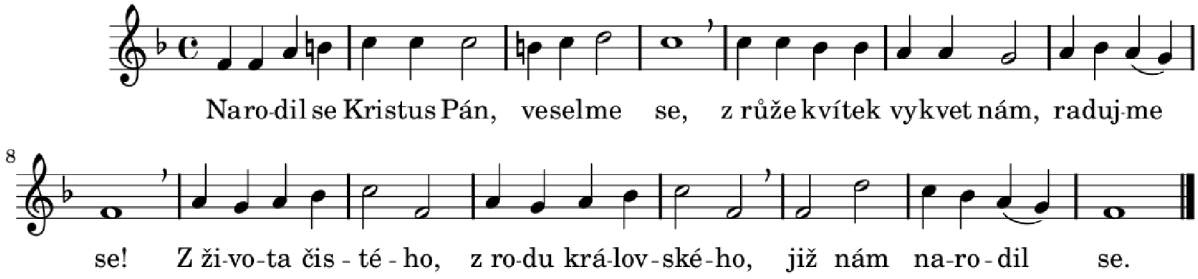
4 Varhanní rejstřík = sada píšťal stejné síly, barvy a charakteru zvuku [5].

5 Manuál = varhanní klaviatura pro ruce, varhany mají obvykle dvě nebo i více těchto klaviatur [5].

6 Žaluzie – v žaluziové skříni stojí obvykle rejstříky celého manuálu, jedná se o dřevěnou skříň, jejíž přední stěna je žaluziově rozdělena, otevíráním a zavíráním žaluzie se zvuk zesiluje a zeslabuje, zavřená žaluzie působí dojmem vzdálenějšího zvuku [5].

Notové ukázky, kterými je prokládán výklad, jsem až na jednu výjimku složil sám. Touto výjimkou je příklad fugata (viz Obr. 2-11 Fugato), při jehož tvorbě jsem vycházel z expozice fugy G dur Josefa Ferdinanda Norberta Segera[7].

K sazbě notového materiálu jsem použil open source program na sazbu not Lilypond, o kterém je řeč v kapitole 3.4.1.



Obr. 2-1 Melodie písně Narodil se Kristus Pán

2.3.1 Tvorba tématu

Pro předehru se obvykle nepoužívá celá melodie písně, ale pouze její část, která je doplněná závěrem. Celá melodie je totiž zbytečně dlouhá a většinou obsahuje příliš mnoho hudebních myšlenek na to, aby se s nimi pracovalo na tak krátkém úseku jako je varhanní předehra. Obvykle se tedy jedná o jeden nebo několik motivů[8], které jsou vybrány z melodie písně. Ty mohou být případně propojeny motivy dalšími, které si varhaník vymyslí sám. Jednotlivé motivy je možné buď citovat doslovně nebo s nimi nějakým způsobem pracovat. Nejčastějšími metodami, které jsou využívány při práci s motivem jsou:

- opakování[9],
- obměňování[10],
- ozdobování[11],
- dělení[12],
- krácení[13],

7 Josef Ferdinand Norbert Seger (1716-1782) byl český hudební skladatel, houslista a varhaník barokního období.

8 Motiv = nejmenší nedělitelný prvek ve skladbě, který má ještě určitý obsah a význam [2].

9 Opakování = nejjednodušší způsob práce s tématem, při němž není změněn ani rytmus ani melodie motivu [2].

10 Obměňování = menší nebo větší obměna melodie (změna velikostí intervalů), rytmu (změna délek rytmických hodnot), melodie i rytmu současně [2].

11 Ozdobování = motiv je obohacen o různé melodické tóny drobných hodnot, takže se jeho délka nemění [2].

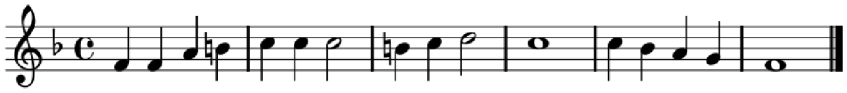
12 Dělení = z motivu se vyjme jen určitá část a s ní se pracuje dále samostatně. Používá se zejména u delších motivů nebo témat [2].

13 Krácení (zkrácení) = vypouštění některých tónů nebo zkracování jejich délek. Celková délka motivu se tedy zkracuje [2].

- rozšíření[14],
- převrat[15],
- račí postup[16].

V hudební praxi se mohou vyskytnout jednotlivé druhy práce s motivem v čisté podobě samostatně, častěji však ve vzájemných kombinacích [2]. Práci s motivy je vhodné při tvorbě předeher využívat, protože téma předehry tím získá na zajímavosti a posluchačské přitažlivosti. Posluchači se potom sami sebe ptají, zda se opravdu jedná o naznačenou píseň. Přesto je však třeba používat tyto kompoziční prostředky střízlivým způsobem. Motivická práce může totiž vést k tak zásadním změnám tématu, že toto už ani vzdáleně nepřipomíná původní píseň. Pro ujištění posluchačů je vhodné v rámci posledního motivu tématu doslovně citovat danou píseň. Nyní již posluchači dobře vědí, o jakou píseň se jedná, mohou tedy začít zpívat.

Příklad tématu předehry je zobrazen na obrázku níže. Pro jednoduchost jsem nevyužil žádné metody práce s motivem. S tímto tématem budu dále pracovat během výkladu v kapitolách, které se zabývají harmonizací tématu a následnou aplikací daného stylu předehry.



Obr. 2-2 Příklad tématu předehry

2.3.2 Harmonizace tématu

Harmonie popisuje, třídí, vysvětluje a hodnotí strukturu akordů (tvar, zápis, funkce, napětí...) a jejich vzájemné vztahy, řazení a spojování v harmonické větě. Harmonizace melodie je potom vlastně proces vyhledávání skryté harmonie, kterou melodie obsahuje [3].

Materiál pro harmonizaci tvoří akordy, které vznikly během dlouhého hudebního vývoje od starověku až po současnost. Stavba akordů je obvykle terciová, což znamená, že akord v základním tvaru se skládá ze dvou nebo více tercií postavených nad sebou. Existují sice i systémy postavené na jiných intervalech (například na kvartě), v rámci této práce jsem se však omezil na akordy terciové stavby.

Přiřazování jednotlivých akordů k odpovídajícím tónům melodie se neděje nahodile, ale řídí se velkým množstvím pravidel a podmínek, které by měly být splněny. Některá pravidla mohou být velmi jednoduchá, jiná zase mnohem složitější. Skladatel si může také určit svá vlastní pravidla nebo se dokonce může rozhodnout, že některá obecně uznávaná pravidla bude záměrně porušovat. Tímto způsobem vznikají styly pro harmonizaci, které jsou typické jak pro jednotlivá období hudebního vývoje, tak pro jednotlivé skladatele. Výsledkem toho všeho je, že harmonizace melodie má takřka nekonečné množství řešení. Danou melodii by pravděpodobně zharmonizoval jinak Johan Sebastian

14 Rozšíření (prodloužení) – vzniká přidáváním několika tónů nebo prodlužováním délek již existujících tónů. Celková délka motivu se tedy prodlužuje [2].

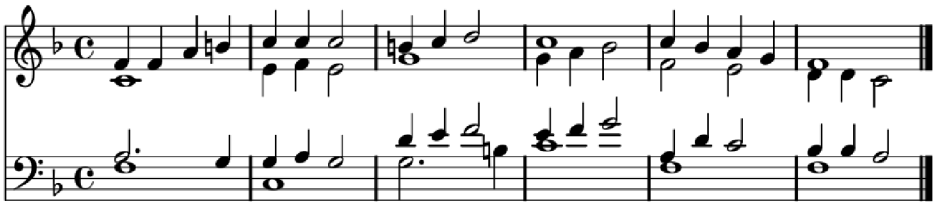
15 Převrat (inverze, obrat, protipohyb) – stoupající intervaly motivu nebo tématu se mění na klesající a naopak. Převrat může být buď přísný (intervaly jsou zachovány ve své velikosti) nebo volný (velikost jednotlivých intervalů se mění, opačný směr pohybu melodické linky však zůstává zachován) [2].

16 Račí postup = uvádění melodie, rytmu nebo obojího pozpátku [2].

Bach než třeba Wolfgang Amadeus Mozart nebo Ludwig van Beethoven. Navíc by každý z těchto velikánů byl schopen vyplodit desítky různých smysluplných řešení tohoto problému.

Je tedy zřejmé, že proces harmonizace melodie je problém dosti složitý, a proto nevede k jednoznačnému řešení. Jedná se o problematiku, o které bylo napsáno mnoho knih, učebnic, článků a skript. Harmonií se potom zabývá mnoho předmětů, které jsou vyučovány na konzervatořích i vysokých školách s hudebním zaměřením.

Jak má tedy varhaník s průměrným hudebním vzděláním postupovat při harmonizaci tématu své předehry? Krása harmonizace spočívá v tom, že je možno k celému problému přistupovat na různé úrovni složitosti. Nejjednodušší doprovody církevních písní většinou vycházejí z několika akordů, a přesto jsou velmi prakticky a funkčně napsané. Je také možné se opřít o doprovod dané písně, který má varhaník obvykle k dispozici. Problém tedy nemusí být tak složitý, jak by se na první pohled mohlo zdát. Příklad jednoduché harmonizace melodie je uveden na obrázku 2-3.



Obr. 2-3 Harmonické schéma

V rámci této práce jsem se rozhodl přistoupit k harmonizaci pomocí předem definovaných pravidel, která se můj systém snaží v co největší míře splnit. O tomto přístupu je podrobněji pojednáno v kapitole 3.3.

2.3.3 Homofonní styly varhanních předeher

S vytvořeným harmonickým schématem lze dále pracovat na základě znalosti stylů neboli forem pro tvorbu varhanních předeher homofonního stylu. V následujících odstavcích budou představeny některé tyto styly. Forem pro předehru je však mnohem více. Je také možné si vytvářet své vlastní. Pro základní představu o dané problematice jsem však vybral pouze několik z nich:

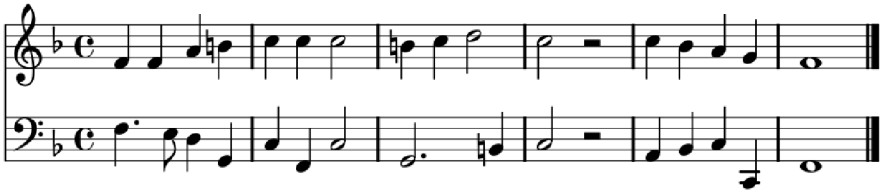
- duo,
- responsoriální přístup,
- rytmizace harmonické sazby,
- melodie v různých hlasech,
- echo.

Duo

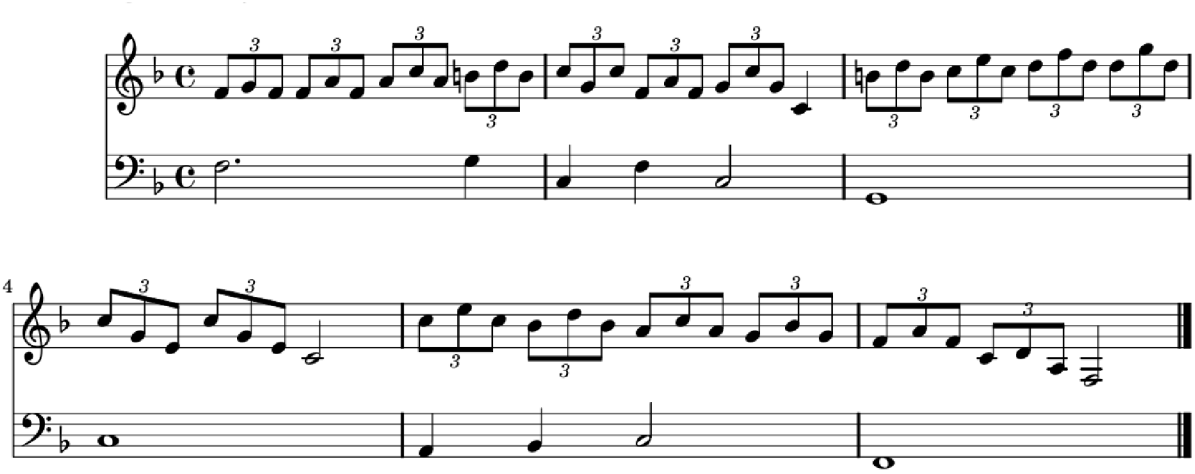
Duo je dvouhlasá forma předehry. Téma předehry se nachází ve výše položeném hlase, kterému říkáme soprán. Doprovodný spodní hlas se nazývá bas. Basovou linku lze získat z harmonického schématu úpravou nejspodnějšího hlasu. Oba hlasy můžeme potom dále figurovat[17]. Pokud tímto

17 Figurace hlasů = výzdoba hlasů melodickými tóny tak, aby se jednotlivé hlasy rytmicky doplňovaly [3].

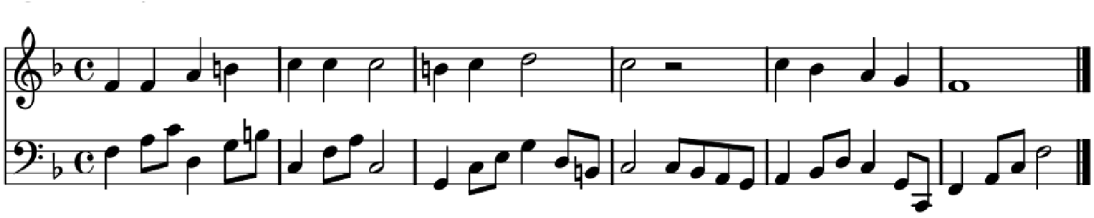
způsobem upravujeme soprán, získáme duo s figurovaným cantem firmem[18]. V opačném případě získáme duo s figurovaným basem.



Obr. 2-4 Duo



Obr. 2-5 Duo s figurovaným cantem firmem



Obr. 2-6 Duo s figurovaným basem

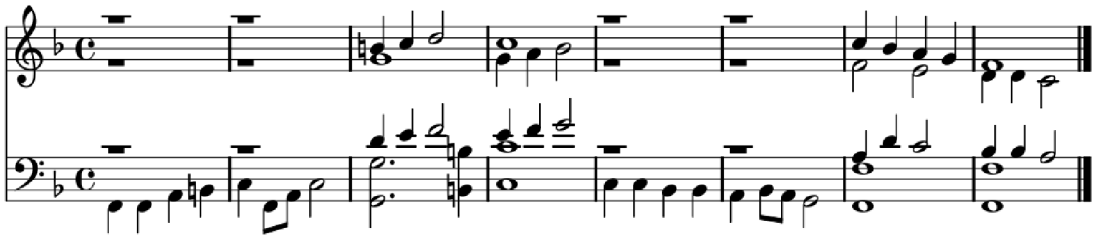
Responsoriální přístup

Jedná se o jednu z nejstarších vícehlasých forem, která byla použita již v Gregoriánském chorálu a dodnes je hojně využívaná například při zpěvu žalmů. Jedná se o jakýsi dialog mezi sólistou a skupinou zpěváků. Dialog je velmi nosný liturgický prvek, který podtrhuje důležitou myšlenku, a to že víra je vztah Boha a člověka a tento vztah se uvádí v život právě rozhovorem.

V intonaci k písni je možno tento princip aplikovat tak, že v jednom sólovém hlase je citována část melodie. Na tu potom navazují ostatní hlasy společně. Sólovým hlasem bývá často bas, protože soprán má svou pevnou pozici jako cantus firmus odpovědi. Bas takto může využít principu kontrastu vůči sopránu. Velmi efektní možností je využití pedálu[19] jako sólového hlasu.

18 Cantus firmus = obecně hlas, který obsahuje základní téma skladby (v našem případě se jedná o téma předehry).

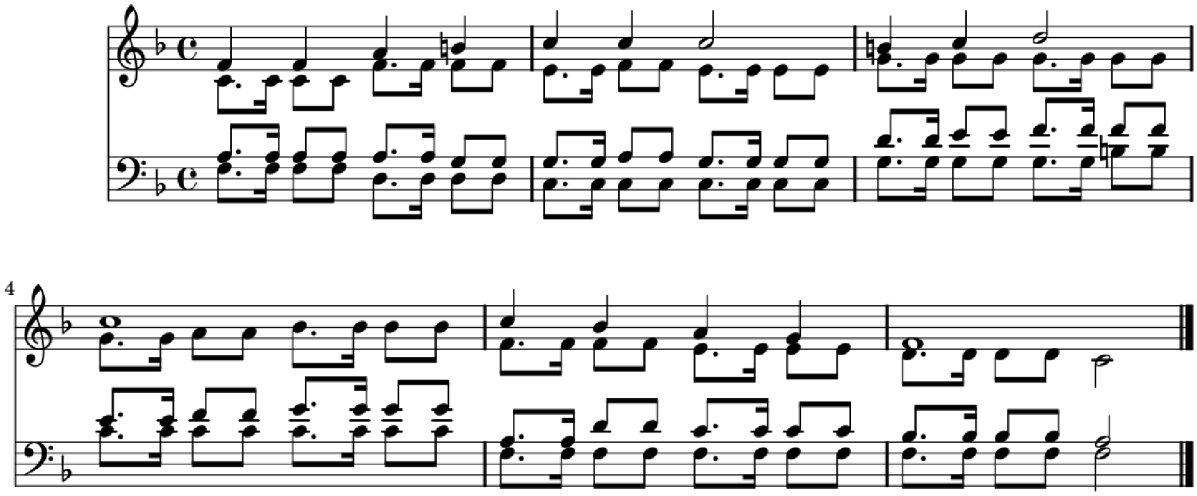
19 Pedál = varhanní klaviatura pro nohy, která je obvykle položena níže než manuál [2].



Obr. 2-7 Responsoriální přístup

Rytmizace sazby

Dalším přístupem je možnost rytmizace sazby harmonického schématu. Cantus firmus zůstává obvykle neměnný. Ostatní hlasy se upraví podle jednoduchého rytmického schématu. V níže uvedeném příkladu, jsem použil tuto rytmickou smyčku: osminová nota s tečkou, šestnáctinová nota a dvě osminové noty. Výsledná úprava má slavnostní charakter a je možno ji hrát i na pleno[20].

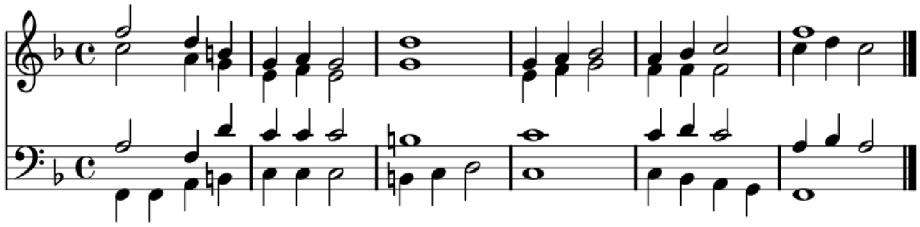


Obr. 2-8 Rytmizace sazby

Melodie v různých hlasech

Cantus firmus nemusí být nutně obsažen v sopránu. Je možno jej umístit i do jakéhokoli jiného hlasu. V následujícím příkladě je téma předehry přiděleno basu. Při interpretaci této předehry je možno využít zvukové odlišnosti jednotlivých manuálů a pedálu. Pedál hraje melodii na silnější rejstříky, tenor v levé ruce je interpretován některým sólovým rejstříkem prvního manuálu a pravá ruka hraje doprovod na slabší zvuk druhého manuálu.

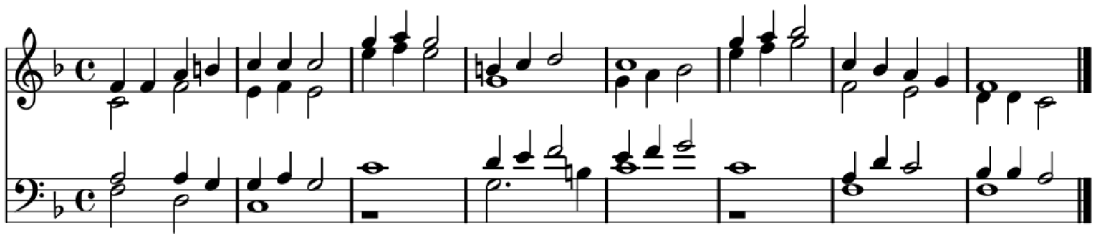
20 Pleno = typický silný varhanní zvuk, u menších varhan je tímto pojmem myšleno použití všech rejstříků.



Obr. 2-9 Melodie v basu

Echo

Poslední uvedenou technikou, která využívá homofonního principu, je echo. Jak již její název napovídá, vychází tento styl z principu ozvěny. Na první manuál je zahrána část melodie písně s doprovodem, její konec je potom opakován v podobě ozvěny na manuálu druhém o oktávu výše.



Obr. 2-10 Echo

2.4 Polyfonní přístup

Polyfonie je způsob skladby založený na dvou nebo více samostatných, současně znějících hlasech. Samostatnost závisí na melodické výraznosti a rytmické rozdílnosti hlasů. Skladba je zpracována na základě melodickém, hudební myšlení je vertikální [4]. Je třeba rozlišovat, že polyfonií se zde nemyslí jakákoli vícehlasá skladba, jak je tento pojem často vnímán. Rozhodujícím faktorem je samostatnost jednotlivých hlasů. Obecný postup při tvorbě předehry s využitím polyfonního přístupu je následující:

- tvorba tématu předehry,
- aplikace daného stylu předehry.

Tvorba tématu předehry vychází ze stejných principů jako je tomu v případě užití homofonního přístupu. O jednotlivých stylech polyfonních předeher je pojednáno níže. Stejně jako u předchozí kapitoly je vše demonstrováno na příkladu písně Narodil se Kristus Pán.

2.4.1 Polyfonní styly varhanních předeher

S vytvořeným tématem lze potom dále pracovat na základě znalosti stylů neboli forem pro tvorbu varhanních předeher polyfonního stylu. Z metod, které využívají polyfonního přístupu, uvádím opět pouze některé zástupce:

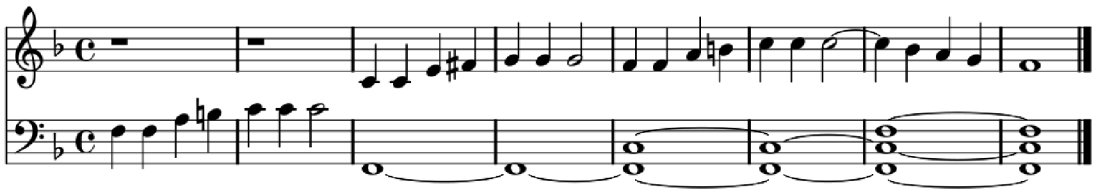
- imitace,
- fugato,

- ostinato,
- melodie na prodlevě.

Stejně jako v případě homofonních stylů je počet možností daleko rozmanitější. První dva zmíněné styly využívají imitaci, která je typická pro polyfonní práci s tématem. Druhé dva se nacházejí někde na pomezí mezi homofonií a polyfonií. Vzhledem k tomu, že nevycházejí z harmonického schématu, zařadil jsem je mezi polyfonní metody.

Imitace

V případě této formy dochází k využití nejjednoduššího imitačního principu. Základem je krátké téma, které je většinou reprezentováno prvním nebo prvními dvěma motivy melodie písně. Dané téma nastupuje v jednom hlase. Po svém skončení je imitováno druhým a následně i třetím hlasem. Imitace jsou obvykle prováděny s nějakým intervalovým posunem (často se jedná o kvintu nebo oktávu).

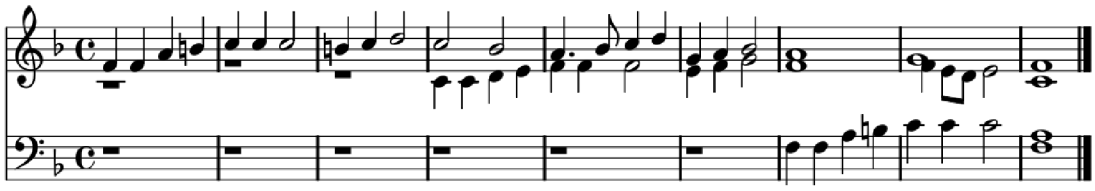


Obr. 2-11 Jednoduchá imitace

Fugato

Slovo fugato znamená fugovaně, tedy na způsob fugy. Značí hlasové nástupy ve stylu fugové expozice, často i v uvolněném tonálním plánu. Není samostatnou hudební formou, nýbrž představuje skupinu imitačních postupů, které se mohou vyskytovat v jednotlivých částech jakýchkoli forem např. v gize, v druhém dílu francouzské ouvertury, v rychlých větách barokních koncertů apod. [4].

Jedná se tedy stejně jako v předchozím případě o použití jednoduchého imitačního principu. Téma je zvoleno podobným způsobem jako v případě imitace. Imitující hlasy nastupují často v kvintách nebo oktávách vzhledem ke hlavnímu tématu.



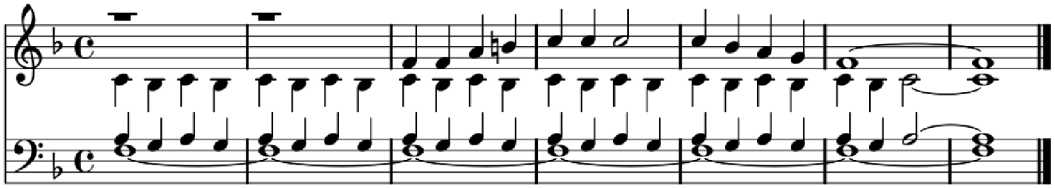
Obr. 2-12 Fugato

Ostinato

Slovo ostinato znamená v hudbě stálé opakování tématu. Z principu opakování vychází rovněž tato forma varhanní předehry. Doprovodné hlasy vytváří jednoduchou ostinátní figuru[21], která se opakuje

21 Ostinátní figura = vícekráte se opakující krátký melodicko-rytmický útvar s menší závažností než motiv, v ostatních hlasech se přitom mění harmonie [3].

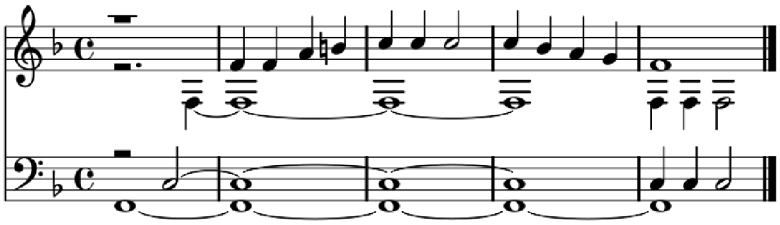
stále dokola. Cantus firmus potom může pouze citovat téma předehry, případně může být toto téma i figurováno nebo jinak upravováno.



Obr. 2-13 Ostinato

Melodie na prodlevě

Posledním z uvedených polyfonních stylů je melodie na prodlevě. Jedná se o velice primitivní styl varhanní předehry. V doprovodných hlasech zní po celou dobu předehry prodleva[22]. V sopránu je potom podobně jako v případě ostinata citováno téma předehry, které může být rovněž figurováno nebo jiným způsobem upravováno.



Obr. 2-14 Melodie na prodlevě

22 Prodleva = delší dobu znějící tón nebo tóny, zatímco v ostatních hlasech se mění harmonie [3].

3 Pohled informatiky

Na předchozích stránkách jsem se zabýval hudební stránkou věci. Tato kapitola má oproti tomu za úkol přiblížit čtenáři, jak se dá s hudbou pracovat pomocí počítačů. Studium této problematiky bylo pro mě velmi důležité, protože mi pomohlo nalézt prostředky, které jsem dále využil při návrhu a implementaci programové části mé práce.

Zabýval jsem se zejména způsoby, které již byly využity pro automatickou kompozici, dále problémem reprezentace not v počítačích, úlohami s omezujícími podmínkami a hledáním vhodného systému pro tvorbu notového výstupu mého systému.

3.1 Přístupy k automatické tvorbě hudby

Pro automatickou kompozici hudby byly v minulosti uplatněny systémy, které jsou založené na různých přístupech. V následujících odstavcích jsem se pokusil vytvořit stručný přehled toho, jak lze danou problematiku řešit. Cílem této kapitoly není představit všechny možné cesty, kterými je možné se ubírat. Rozhodl jsem se vybrat pouze několik základních přístupů, abych ukázal, že možností je značné množství. Tyto přístupy jsem rozdělil do tří skupin, kterým jsou věnovány následující podkapitoly této práce. Jedná se o tyto přístupy:

- systémy založené na pravidlech,
- systémy založené na náhodě,
- systémy založené na učení.

3.1.1 Systémy založené na pravidlech

Tento přístup vychází z faktu, že hudbu lze popsat pomocí pravidel, které je třeba v určité míře dodržovat. K řešení problematiky se potom nabízí využít úloh s omezujícími podmínkami.

Vzhledem k tomu, že jsem se rozhodl tento přístup aplikovat v rámci své práce, je mu věnována celá kapitola 3.3.

3.1.2 Systémy založené na náhodě

Princip využití náhody pro tvorbu hudby není žádnou novinkou. Je znám již delší dobu a je dodnes široce využíván hudebními skladateli. Ve dvacátém století pak dokonce vznikl styl kompozice zvaný aleatorika, který přímo vychází z náhody. Mezi nejznámější průkopníky této hudby patřili Pierre Boulez a John Cage.

Jako příklady systémů, které jsou založeny na tomto principu, uvádím stochastické modely a Markovské řetězce.

Stochastické modely

Jak již název napovídá, jedná se o modely, které vycházejí ze zákonů pravděpodobnosti. Systém se rozhoduje na základě náhodných čísel, a proto je nemožné předpovídat jeho výstup v jakémkoli konkrétním čase v budoucnosti [6].

Systémy jsou většinou zaměřené na experimentování s melodiemi na základě různých uživatelem zadaných parametrů, které řídí proces generování hudby. Mezi takovéto parametry patří například míra tonality, tónina, tempo, rytmus rozsah použitých not, koeficient opakování not, maximální délka fráze a jiné [7].

Markovské řetězce

Jiný přístup k využití pravděpodobnosti mají Markovské procesy. Jedná se o diskrétní systémy, pro které platí, že pravděpodobnost přechodu do budoucího stavu závisí jen na aktuálním stavu systému. Jednotlivé stavy jsou tedy na sobě sekvenčně závislé [6]. Tento fakt vede k použití tzv. Makovských řetězců.

Při použití těchto řetězců je skladba popsána jako sled různých stavů, přičemž jednotlivé stavy mohou být reprezentovány různými veličinami (výška noty, délka noty, síla úderu apod.). Pro každý stav je potom vytvořen vektor přechodových funkcí P(A, B) s pravděpodobností přechodu ze stavu A do stavu B [7]. Spojením těchto vektorů může potom vzniknout přechodová matice nebo orientovaný graf s ohodnocenými hranami. Oba tyto způsoby reprezentace potom mohou sloužit jako stochastický model pro generování posloupnosti tónů s danými pravděpodobnostmi přechodu z daného stavu do jiného.

3.1.3 Systémy založené na učení

Tento přístup spočívá především v přípravě trénovacích dat pro nástroj umožňující strojové učení. S použitím tohoto nástroje se trénováním vytvoří model, který se jednorázově použije pro harmonizaci. Úspěch harmonizace pak především závisí na vhodně zvolených atributech, které se pro trénování využijí. Prvotním klíčem k úspěchu je pak zvolení vhodných trénovacích dat [8].

Zajímavou výhodou tohoto přístupu je možnost ovlivnit hudební styl, v jakém je harmonizace provedena. Pokud bychom například systém trénovali na Bachových skladbách, mělo by potom být možné generovat výstupy, které odpovídají Bachovu stylu harmonizace.

3.1.4 Další systémy

Dalšími nástroji, které je možno pro automatickou tvorbu hudby využít mohou být například: celulární automaty, gramatiky, fraktály apod.

3.2 Reprezentace not v počítačích

Reprezentace hudby v počítačích slouží k zachování informací spojených s hudbou. Existuje mnoho formátů, které jsou většinou určeny ke speciálním účelům. Příkladem může být například hudba určená k poslechu ve formátu mp3 nebo běžný notový zápis v grafické podobě. V následující kapitole se však budu zabývat systémy symbolických reprezentací pro počítačové programy. Jedná se tedy například o struktury pro reprezentaci notového zápisu.

3.2.1 Základní přístupy

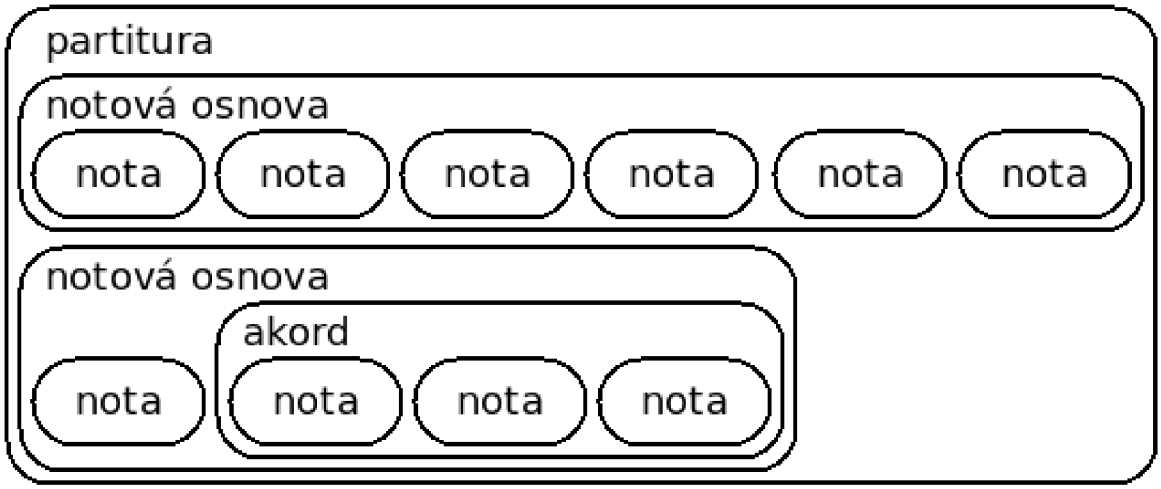
Nejzákladnějším přístupem je reprezentace not založená na událostech. Tento přístup je použit například u MIDI souborů. Událost je reprezentována pomocí sady parametrů. Každý parametr potom reprezentuje nějakou vlastnost notového zápisu (např. délku tónu, frekvenci tónu, rychlost úhozu nebo čas, kdy tón zazní atd.). Notový zápis je potom reprezentován pomocí seznamu událostí.

Jiným přístupem je potom parametrická reprezentace notového zápisu, kdy jednotlivé parametry z předchozího příkladu mohou být reprezentovány odlišnými způsoby. Typickým využitím tohoto přístupu může být možnost rozlišit, zda se jedná o notu cis nebo des. Pokud bychom tyto noty zapsali pomocí jejich frekvencí, ztratili bychom informaci o tom, o kterou z nich se jedná. Jiným příkladem může být časové určení délky tónů v milisekundách nebo v dobách.

3.2.2 Hierarchická reprezentace

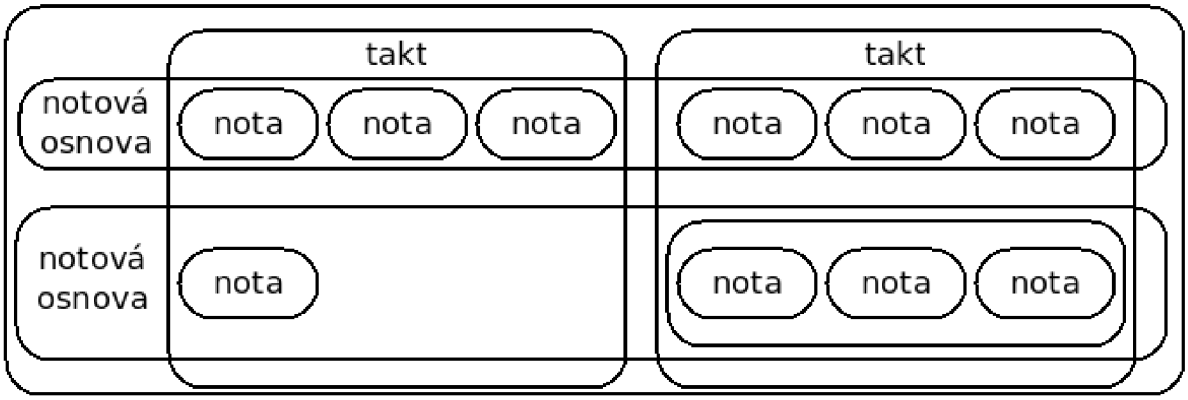
Tento přístup vychází z faktu, že na notovou partituru lze aplikovat princip dekompozice a rozdělit ji tak na jednotlivé objekty (např. akordy, motivy, témata, jednotlivé hlasy atd.). Tyto části partitury mohou být potom hierarchicky tříděny a uspořádány. Notový zápis je potom reprezentován pomocí množiny kontejnerů, které obsahují patřičné objekty.

Jedním ze způsobů, jak takto vzniklé kontejnery lze uspořádat, může být vytvoření stromové struktury. Tento přístup je široce rozšířen zejména u jazyků systémů pro sazbu not typu LilyPond apod.



Obr. 3-2 Uspořádání kontejnerů do stromové struktury

Jiným způsobem může být uspořádání kontejnerů do acyklického grafu. Tento přístup vychází z faktu, že na jednotlivé objekty lze v témže čase logicky nahlížet z různých úhlů pohledu (např. jednotlivé noty mohou být součástí určitého motivu, akordu i taktu zároveň). Z takovéto reprezentace lze potom určit více informací, které logickým způsobem souvisí s daným objektem.

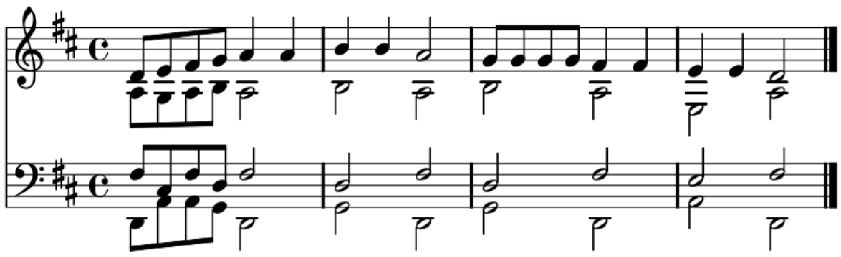


Obr. 3-2 Uspořádání kontejnerů do acyklického grafu

3.3 Úlohy s omezujícími podmínkami

V programové části svého diplomového projektu jsem řešil problémy, které jsou úzce spojeny s problematikou úloh s omezujícími podmínkami (Constraint Satisfaction Problem – CSP). Je tomu tak z toho důvodu, že na hudbu je možno se dívat do jisté míry z pohledu matematiky. Například harmonizace melodie je popsatelná konečnou množinou pravidel, které je třeba při vytváření jednotlivých doprovodných hlasů dodržovat. Právě v případě harmonizace jsem se rozhodl využít principů řešení omezujících podmínek. Harmonizaci jsem totiž popsal pomocí pravidel, kterých je nemalé množství. Bez účinných nástrojů a algoritmů by bylo prakticky nemožné ošetřovat platnost všech těchto pravidel.

Nejedná se o jediný možný přístup, který lze při řešení problémů automatické kompozice hudby použít. Například ve své bakalářské práci Harmonizace melodie jsem se obešel i bez metod řešení CSP. To mi umožnilo úspěšně generovat jednoduchou harmonizaci k zadané vstupní melodii. Ukázalo se však, že systém podléhal celé řadě omezení. Není možné žádným způsobem ovlivnit výstup programu, systém nedokáže vytvořit harmonizaci k melodii, která obsahuje jiné než doškální tóny[23] apod. Proto jsem se rozhodl použít nějaký silnější nástroj, než jsou vlastnoručně zhotovené algoritmy. Tímto nástrojem jsou úlohy s omezujícími podmínkami a využití knihovny Gecode pro programovací jazyk C++.



Obr. 3-2 Ukázka výstupu bakalářské práce Harmonizace melodie

23 Doškální tón = tón, který se nachází v dané stupnici.

3.3.1 Definice CSP

Pokud chceme definovat pojem CSP, je vhodné se zamyslet nad tím, co to vlastně je omezující podmínka. Jedná se o relaci zachycující vztah mezi proměnnými, které tato obsahuje. Cílem systémů řešících omezující podmínky je potom nalézt takové ohodnocení proměnných, pro které platí, že všechny podmínky jsou splněny [10].
Formálně je potom úloha s omezujícími podmínkami definována takto:

- Nechť je dána množina proměnných $\{X_1, X_2 \dots X_n\}$, z nichž každá může nabývat hodnot neprázdné domény D_i.
- Nechť existuje množina omezujících podmínek $\{C_1, C_2 \dots C_m\}$, z nichž každá podmínka předepisuje vztah mezi nějakou podmnožinou proměnných z výše zmíněné množiny proměnných.
- Nechť stav úlohy je definován hodnotami přiřazenými jednotlivým proměnným a nechť legální stav znamená, že proměnné s přiřazenými hodnotami vyhovují předepsaným podmínkám.
- V počátečním stavu není přiřazena hodnota žádné proměnné.
- Obecný operátor přiřazuje hodnotu libovolné volné proměnné tak, aby následující stav byl legálním stavem.
- Řešením úlohy je legální stav, ve kterém mají všechny proměnné přiřazené hodnoty [11].

3.3.2 Jednoduché úlohy řešitelné pomocí CSP

Typickými příklady jednoduchých úloh s omezujícími podmínkami je problém N-dam a řešení rovnice SEND + MORE = MONEY. V prvním případě se snažíme rozmístit na šachovnici o straně n odpovídající počet dam tak, aby se žádné dvě vzájemně neohrožovaly. V druhém případě se jedná o řešení krypto-aritmetického hlavolamu, kdy za jednotlivá písmena dosazujeme různé číslice tak, aby rovnice byla pravdivá.

Pokud chceme vyřešit jakoukoli úlohu založenou na problematice CSP, je třeba ji nejprve řádně formulovat. Na tomto kroku často závisí úspěch našeho řešení.
Formulace úlohy N-dam (předpokládáme, že dámy se nacházejí různých sloupcích):

- množina proměnných $\{D_1, D_2 \dots D_N\}$,
- domény jsou stejné pro všechny proměnné {1, 2 … N},
- omezující podmínky:
  1. Žádné dvě dámy se neohrožují:
     $\forall i \neq j: D_i \neq D_j \wedge |i-j| \neq |D_i - D_j|$

Formulace úlohy SEND + MORE = MONEY:

- množina proměnných {S, E, N, D, M, O, R, Y},
- domény jsou stejné pro všechny proměnné {0, 1 ... 9},
- omezující podmínky:
  1. čísla nesmí začínat nulou,
  2. každé písmeno reprezentuje odlišnou číslici,
  3. následující lineární rovnice musí být řešitelná:

```
              S*1000 + E*100 + N*10 + D
+             M*1000 + O*100 + R*10 + E
= M*10000 + O*1000 + N*100 + R*10 + Y
```

3.3.3 Metody řešení CSP

CSP lze řešit pomocí metod prohledávání stavového prostoru (lze použít všechny informované i neinformované metody). Princip tohoto řešení je následující: počátečním stavem je prázdné přiřazení. V jednotlivých krocích potom přiřazujeme hodnoty do volné proměnné tak, abychom neporušili žádnou podmínku. Řešením je potom úplné konzistentní přiřazení. Existují však i efektivnější metody řešení úloh s podmínkami.

V následujícím textu budou stručně popsány dvě základní metody řešení CSP: backtracking a forward checking. Sofistikovanější metody, které se často používají v praxi, vycházejí zpravidla právě z těchto dvou základních metod.

Backtracking

Backtracking, neboli prohledávání s navracením, je jedním z nejrozšířenějších algoritmů pro řešení CSP. Jedná se o základní neinformovaný algoritmus.

V jednotlivých krocích jsou přiřazovány hodnoty do volných proměnných. V případě, že algoritmus narazí na nekonzistenci, vrátí se k poslední ohodnocené proměnné a přiřadí jí jinou hodnotu. Pokud nastane takový případ, že dané proměnné již nelze přiřadit žádnou hodnotu z její domény, vrací se algoritmus k předchozí proměnné.

Tento algoritmus však může být často velmi neefektivní. Proto byly vyvinuty metody, které se snaží zvýšit efektivitu prohledávání. Velikost prohledávaného stavového prostoru může být zmenšena například díky znalosti struktury daného problému nebo pořadím proměnných, kterým jsou přiřazovány hodnoty.

Často používanou heuristikou je princip první chyby (first fail), kdy se snažíme nejprve vyřadit ty možnosti, které k řešení problému nevedou. Z tohoto úhlu pohledu můžeme vybrat nejvíce omezenou proměnnou. Tedy proměnnou s nejmenší doménou (doménová heuristika). Druhou možností je výběr proměnné s nejvíce podmínkami.

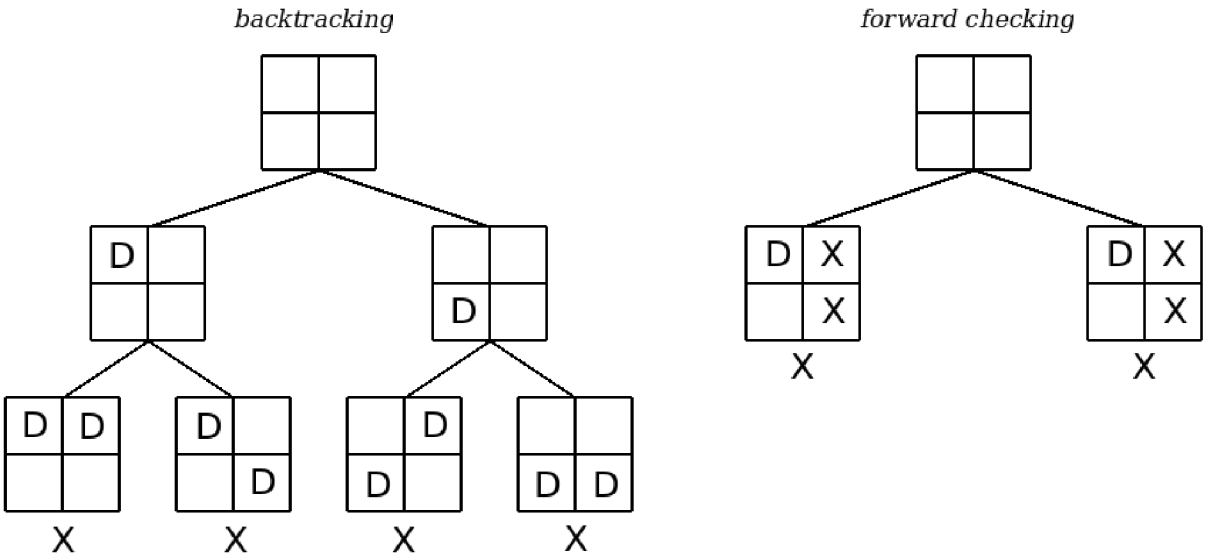
Jinou heuristikou je potom princip prvního úspěchu (suceed first), kdy se naopak snažíme preferovat hodnotu, která nejspíše patří do řešení. Touto hodnotou bývá často takové přiřazení, které nejméně omezí zbývající volné proměnné.

Forward checking

Forward checking, neboli dopředná kontrola, je označení skupiny algoritmů, které se snaží předejít konfliktu při prohledávání.

Princip algoritmu spočívá v tom, že se po přiřazení hodnoty dané proměnné kontrolují všechny podmínky a hodnoty proměnných, které dosud nebyly ohodnoceny (tzv. budoucí proměnné). Touto kontrolou je možné omezovat domény budoucích proměnných. Pokud má některá z budoucích proměnných prázdnou doménu, je zřejmé, že současné částečné řešení je nekonzistentní. Tímto způsobem je potom možno odhalit větve vedoucí k neúspěchu dříve než u metody backtracking.

Následující obrázek znázorňuje princip fungování metod backtracking a forward checking v jejich základní podobě. Jako demonstrační příklad poslouží problém dvou dam. Budeme předpokládat, že každá dáma se nachází v jiném sloupci. Každá dáma je tedy reprezentována doménou {1, 2}, která reprezentuje čísla řádků. Z obrázku je dobře vidět, že pro hodnotu N=2, je úloha N-dam neřešitelná.



Obr. 3-3 Princip fungování metod backtracking a forward checking

3.3.4 Gecode

Gecode je otevřené, volné, přenositelné, dostupné a efektivní prostředí pro vývoj systémů a aplikací založených na CSP [12]. Jedná se o knihovnu implementovanou v C++, která odpovídá standardům tohoto jazyka, a proto může být zkompilovaná běžnými překladači pro tento jazyk na různých platformách. Tato knihovna je využívána rozsáhlou komunitou uživatelů, z toho důvodu je stále dále vyvíjena a zdokonalována.

První verze knihovny byla vydána v prosinci roku 2005. Od té doby se tato knihovna stihla natolik rozšířit, že se v současnosti jedná o jeden z nejpoužívanějších systémů pro řešení CSP. Rozšíření této knihovny je způsobeno zejména její kvalitou, volnou dostupností, rychlostí zpracování problémů a platformní nezávislostí.

Tuto knihovnu jsem pro řešení CSP zaměřených na tvorbu varhanních předeher zvolil z několika důvodů. Za prvé se jedná o systém napsaný v C++, což je efektivně pracující programovací jazyk, který je mi známý. Dalšími výhodami jsou přenositelnost, kvalitní dokumentace v podobě mnoha komentovaných příkladů a v neposlední řadě také to, že se jedná o volně dostupné vývojové prostředí.

3.3.5 Strasheela

Díky vedoucímu své práce Ing. Michalu Fapšovi jsem se seznámil s projektem Strasheela. Jedná se o práci Anderse Torstena, jejímž výsledkem je kompoziční systém založený na omezujících podmínkách.

Anders Torsten se narodil roku 1968 v Güstrow (Německo). Studoval muzikologii a teologii na Humboldtské univerzitě v Berlíně. Později se specializoval na elektroakustiku. Ve svých výzkumech se zabývá možností využití počítačů při komponování hudby. To bylo motivací k vyvinutí Strasheely jakožto programu, který využívá umělou inteligenci ke kompozici hudby [13].

Torsten vyšel z předpokladu, že hudební kompozici lze popsat souhrnem matematických pravidel, která je třeba dodržovat. Tato pravidla lze reprezentovat pomocí úloh s omezujícími podmínkami.

Výhodou systému Strasheela je to, že nabízí uživateli možnost definice vlastních pravidel pro CSP. Touto cestou může skladatel dosáhnout výsledků, které přesně odpovídají jeho představám. Je tedy možno vytvářet hudbu podobným způsobem jako třeba v baroku. Jinou cestou je definování vlastních stylů. Tato pravidla si však musí uživatel definovat pomocí programovacího jazyka Oz. To je jistá nevýhoda, protože pokud uživatel není programátor, je pro něj velmi obtížné Strasheelu používat. Největší výhoda systému je zároveň paradoxně také jeho největší nevýhodou.

Strasheela je tedy komplexní a robustní software pro automatizovanou tvorbu hudby. Z pohledu běžného uživatele se však jedná o systém, který není jednoduché zvládnout.

3.4 Notační programy

V dnešní době existuje poměrně velké množství programů na psaní not. Jejich hlavním účelem je sazba klasického notového zápisu tak, aby tento byl dobře čitelný a interpretovatelný živými hudebníky. V tomto ohledu se tedy notační programy podobají specializovaným textovým procesorům. V dnešní době se však již funkcionalita pokročilejších z těchto programů částečně překrývá s možnostmi digital audio workstation[24] [15].

V rámci svého projektu jsem hledal takový notační program, který by byl schopen pracovat s textovým vstupem. Díky takovému programu by potom bylo možné vytvářet výstup celého systému v podobě notového zápisu. Jako ideální program pro tyto účely se ukázal být LilyPond.

3.4.1 LilyPond

LilyPond je open source program vytvořený v jazyce C++, který se specializuje na sazbu notového zápisu. Tento program je schopen produkovat výstup na takové úrovni, že je možné jej použít i jako profesionální software pro sazbu not (jako je např. Sibelius, Finale, Capella apod.). Jeho velkou výhodou je i to, že je dostupný v distribucích pro různé operační systémy. Stačí si jej pouze stáhnout z domovské stránky projektu a nainstalovat. Funguje na podobném principu jako TeX. Na vstupu je prostý textový soubor, ten je přeložen a zpracován programem. Výstupem může být například soubor ve formátu pdf, který obsahuje tisknutelnou formu notového zápisu.

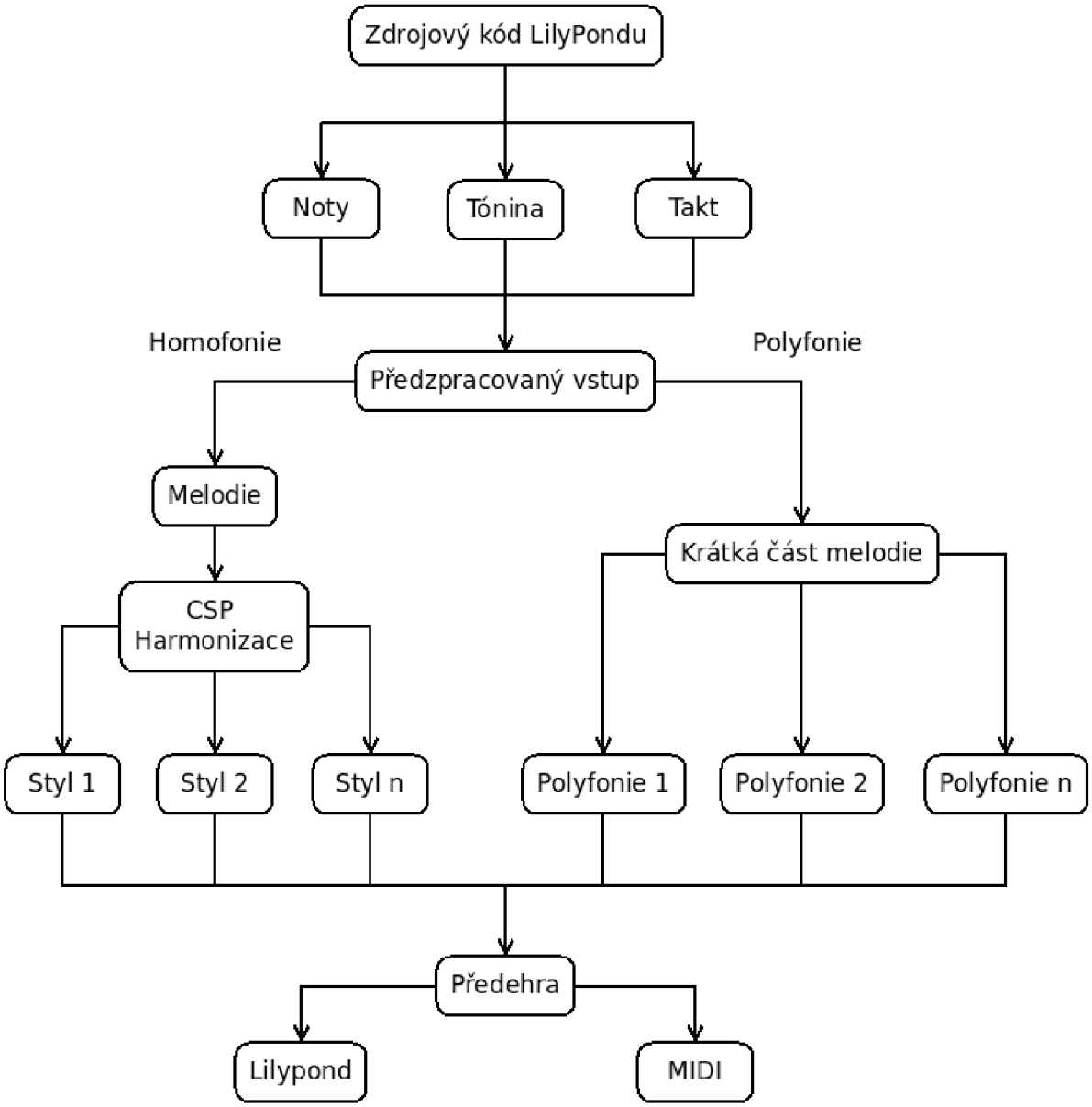
Možnost využití textového vstupu a volná dostupnost jsou rozhodujícími podmínkami pro využití této aplikace v rámci mé práce. Jediným problémem, který budu muset řešit, je tedy práce se zdrojovým kódem pro LilyPond. Vše ostatní obstará LilyPond sám.

24 Digital audio workstation (DAW) = kompletní řešení pro produkci hudby a zvuku pomocí počítače. Mají potenciál nahradit i celé hardwarové vybavení hudebního studia [15].

4 Návrh systému

Mým cílem bylo navrhnout jednoduchou konzolovou aplikaci, která by na základě textového vstupu umožňovala vytvořit jednoduchou varhanní předehru. Pro konzolovou aplikaci jsem se rozhodl z toho důvodu, abych se mohl soustředit výhradně na jádro problému a nemusel se zabývat vytvářením grafického uživatelského rozhraní, které by mohlo zbytečně komplikovat přenositelnost systému. Navíc je možné jej vytvořit dodatečně.

Návrh systému je znázorněn následujícím schématem (viz Obr. 4-1). Při tvorbě projektu jsem předpokládal použití jazyka C++. Pro řešení problémů spojených s harmonizací jsem se rozhodl využít knihovnu Gecode, o které byla řeč výše. Jako externí program pro práci se vstupem a výstupem jsem se rozhodl použít LilyPond.



Obr. 4-1 Schéma návrhu systému

Na vstupu je očekávaná melodie církevní písně v podobě zdrojového souboru pro systém LilyPond. Tento vstup je nejprve třeba předzpracovat. Cílem tohoto předzpracování je extrakce jednotlivých not melodie, tóniny a taktu. Dále je možno aplikovat dva základní přístupy tvorby varhanní předehry, a to buď homofonii nebo polyfonii. Homofonní neboli vertikální přístup vychází z harmonizace melodie. Polyfonní neboli horizontální přístup vede k vytvoření vícehlasé předehry, ve které jsou jednotlivé hlasy více samostatné, přesto však tvoří latentní harmonii. Výstupem programu je potom opět zdrojový kód pro aplikaci LilyPond. Ten lze použít buď k sazbě notového zápisu předehry nebo k vytvoření MIDI výstupu, který může být přehrán uživateli.

4.1 Požadavky na systém

4.1.1 Možnost ovlivnění výstupu uživatelem

Je pravda, že i bez možnosti ovlivnění výstupu uživatelem by bylo možno vytvářet kvalitní a zajímavé předehry. Zajímavějších výsledků je však možno dosáhnout díky interakci s uživatelem. Ten potom může se systémem různým způsobem experimentovat a získat tak výsledek, který lépe odpovídá jeho představám. Uživatel potom může poskytovat lepší zpětnou vazbu tvůrci projektu. Další nespornou výhodou těchto experimentů je možnost využít systém k výukovým účelům. V tomto případě si může uživatel jednoduše prakticky vyzkoušet, co se stane, aplikují-li se určitá pravidla. Ideální systém by potom poskytoval jednoduché intuitivní rozhraní pro tvorbu vlastních pravidel harmonizace i stylů předeher.

Je tedy otázkou, co všechno by měl mít uživatel možnost ovlivnit a jakým způsobem to provádět. Vzhledem k tomu, že se jedná o konzolovou aplikaci, je pravděpodobně nejlepší použít vstupní parametry nebo konfigurační soubor. V našem případě by se jednalo o velké množství vstupních parametrů, proto jsem se rozhodl pro variantu konfiguračního souboru.

Prostřednictvím tohoto souboru je možné ovlivňovat pravidla pro tvorbu tématu předehry, pravidla pro harmonizaci a vybrat styl předehry, který bude pro danou píseň aplikován. Vstupní parametry programu jsou potom určeny pro volbu vstupního či výstupního souboru apod.

4.1.2 Nenáročnost z hlediska uživatele

Zde je třeba se nejprve zamyslet nad tím, kdo by mohl být koncovým uživatelem navrhovaného systému. Pravděpodobně se nebude jednat o zdatného informatika, který by byl schopen editovat zdrojový kód a měnit si tak systém dle vlastních představ. Mezi varhaníky jsou sice takoví lidé, ale není jich mnoho a divil bych se, kdyby se jim chtělo něco takového dělat. Uživatel však nebude ani skladatel profesionál, ten je totiž schopen si jednoduchou předehru vytvořit ve formě improvizace sám. Předpokládejme tedy, že uživatel bude průměrně vzdělaný varhaník s natolik pozitivním vztahem k počítači, že je ochoten zkusit použít automatický systém pro tvorbu předeher a věří, že by mu tento systém mohl být v něčem nápomocen.

Takového uživatele je třeba neodradit zbytečnými složitostmi. Myslím, že editace konfiguračního souboru není věc příliš náročná, pokud má uživatel k dispozici vzorový konfigurační soubor a návod, jak s takovým souborem pracovat. Trochu méně pohodlné je vytváření vstupu programu. Je otázkou, zda existuje nějaká jednodušší cesta. Zajímavým způsobem, jak zjednodušit

uživateli práci se systémem, by mohlo být použití MIDI vstupu. Ten sice není zahrnut v původním návrhu aplikace, ale je možno jím tuto aplikaci snadno rozšířit. Vhodným nástrojem pro toto rozšíření se zdá být nástroj midi2ly, který slouží pro převod MIDI na zdrojový kód pro aplikaci LilyPond. Je však třeba jej využít v kombinaci se vstupem z kláves, které mají MIDI rozhraní pro výstup.

4.1.3 Možnost notového výstupu

Pro praktické využití celého systému je třeba nabídnout uživateli možnost notového výstupu. Noty si totiž může uživatel vytisknout, a potom je přehrát na varhany jako předehru k dané písni. Tvorba systému na sázení not je však velice náročná činnost a není podstatou této práce. Proto jsem se rozhodl využít systém LilyPond, o kterém je pojednáno výše. Výstupem programu tedy musí být zdrojový kód pro tento systém.

4.1.4 Možnost zvukového výstupu

Kdyby byl notový záznam jediným výstupem programu, neměl by uživatel představu o tom, jak bude výsledná předehra znít ve skutečnosti, dokud by si ji nepřehrál na varhany nebo jiný hudební nástroj (předchozí tvrzení vychází z předpokladu, že uživatel nemá absolutní hudební sluch). Proto je třeba vytvořit nějaký zvukový výstup, který slouží ke kontrolnímu přehrání předehry.

K tomuto účelu je vhodné vytvořit MIDI soubor, který je možno přehrát prakticky na každém počítači se zvukovým výstupním zařízením. Zvuková kvalita přehrávaného MIDI souboru je závislá na použitém MIDI syntetizéru a na kvalitě zvukové karty počítače. To však to příliš nevadí, protože tento výstup slouží pouze pro představu uživatele o vytvořené předehře. Výhodou MIDI výstupu je jeho snadná tvorba a malá velikost výstupních souborů. MIDI soubor lze snadno vytvořit díky LilyPondu, který je použit i pro tvorbu notového výstupu.

4.2 Návrh modulů

Celou aplikaci jsem se rozhodl navrhnout a implementovat v jazyce C++ s využitím knihovny Gecode pro řešení CSP. K problematice návrhu jsem přistupoval z pohledu strukturovaného programování. Jelikož se jedná o rozsáhlejší projekt, bylo třeba nejprve provést dekompozici problému a rozdělit jej tak na logicky oddělené části. Takto vytvořené části budu nadále nazývat moduly, přičemž každý z nich je reprezentován jedním souborem. Původní verze tohoto návrhu je ve zjednodušené podobě nastíněna v následujících podkapitolách. Přesnějšímu popisu jednotlivých modulů se věnuje následující kapitola popisující implementaci systému.

4.2.1 Hlavní modul

Hlavní modul projektu, který spravuje běh celé aplikace tím, že spouští patřičné funkce jednotlivých modulů, které jsou mu podřízeny.

4.2.2 Modul pro čtení

Vstupem celého systému je melodie církevní písně v podobě zdrojového souboru programu LilyPond. Výhodou tohoto vstupu je jeho snadné zpracování a snadné vytvoření. Uživatel nemusí chodit k varhanám s nahrávací technikou, aby získal vstup pro danou píseň. Stačí mu vytvořit zdrojový soubor pro LilyPond. Pro tento účel bude možno využít šablony a popřípadě i stručného návodu, aby se uživatel nemusel podrobně učit syntaxi programu LilyPond.

Z tohoto vstupu je třeba získat především tyto informace: notový obsah, tóninu, začátky a konce jednotlivých frází a takt. Všechny tyto informace lze vyčíst přímo ze vstupního souboru. Dále je třeba je vhodně reprezentovat v datových strukturách programu. Toto se týká především notového obsahu. U každého tónu je třeba zjistit jeho délku a výšku. Tyto informace pak budou uloženy v příslušné struktuře, která reprezentuje vstupní melodii.

4.2.3 Modul pro vytváření melodie

Při vytváření melodie předehry je třeba vybrat si některou frázi ze vstupní písně a s ní potom dále pracovat. Je možné použít některé z technik pro práci s motivem (opakování, ozdobování, obměňování, apod.). Nakonec je třeba k takto upravené části melodie přidat závěr, který na ni logicky navazuje z hlediska rytmického i z hlediska melodického.

Tato poslední fráze melodie by měla končit na základním tónu stupnice, které přísluší daná píseň. Je tomu tak proto, aby se utvrdila tónina, ve které se bude zpívat. Často se proto pro tyto účely používá poslední fráze vstupní melodie, která ve velké většině případů končí na tónice.

4.2.4 Modul pro harmonizaci

Rozhodl jsem se, že pro kompozici předeher budu vycházet z harmonického schématu. Rozhodně se nejedná o jediný možný přístup. Většina skladatelů tvoří svá díla alespoň částečně pomocí intuice. Pro automatizovanou kompozici je však vhodnější použít přístup, který lze jednodušeji formalizovat. Tímto přístupem je právě možnost vyjít z harmonického schématu, které lze vytvořit pomocí daných pravidel. Při vytváření tohoto schématu lze vyjít z principů úloh s omezujícími podmínkami a využít knihovnu Gecode, o které byla řeč již dříve.

4.2.5 Modul pro vytváření předeher

Pokud máme již vytvořené téma, které jsme zharmonizovali pomocí CSP, je možno na tomto základě stavět dále a aplikovat některý styl pro homofonní předehry. V rámci své práce bych rád implementoval všechny předehry, které jsem uvedl v kapitole 2.3.3 Homofonní styly varhanních předeher. Jedná se tedy o tyto styly: duo, responsoriální přístup, rytmizace harmonické sazby, melodie v různých hlasech a echo. Systém by však měl zároveň být natolik obecný, aby jej bylo možno rozšířit i o další styly.

Na rozdíl od homofonního přístupu se při tvorbě polyfonní předehry nevychází z harmonického schématu, ale všechny hlasy jsou do jisté míry samostatné a mohou fungovat i samy o sobě. Stejně jako v případě tvorby homofonních předeher, chci i u předeher polyfonních aplikovat všechny přístupy, které jsem uvedl v rámci svého teoretického výkladu. Jedná se tedy o styly představené v kapitole 2.4.1 Polyfonní styly varhanních předeher: imitace, fugato, ostinato a melodie na prodlevě.

Rovněž v případě polyfonních předeher je třeba uvažovat tak, aby byl systém dále rozšiřitelný o další styly předeher.

4.2.6 Modul pro výstup

Výstupem systému by měl být zdrojový kód pro aplikaci LilyPond. Tento výstup je třeba vyrobit ze struktury, která reprezentuje vytvořenou předehru. Problematika vytváření příslušného výstupního textového souboru z daných dat je méně složitá než parsování souboru vstupního. Proto si myslím, že by to neměl být zásadní problém.

4.2.7 Pomocné moduly

Systém by měl obsahovat ještě pomocné moduly implementující práci s pamětí, výpis chybových hlášení, zpracování konfiguračního souboru apod.

5 Implementace

Při implementaci jsem se držel svého původního návrhu a použil jsem programovací jazyk C++ s knihovnou Gecode. Dále jsem využil ještě některé další standardní knihovny pro C++ (fstream, string apod.). Využití LilyPondu pro práci se vstupem a výstupem zůstalo rovněž zachováno.

Implementovaný systém je konzolová aplikace. Během svých studií na Fakultě informačních technologií jsem vyvíjel a testoval velké množství jednoduchých i složitějších konzolových aplikací. Zkušenost mi ukázala, že velmi výkonnou platformou s efektivními nástroji pro vývoj těchto aplikací jsou různé distribuce Linuxu. Proto jsem zvolil své oblíbené Ubuntu 9.10 Karmic Koala. Na této platformě jsem projekt vyvíjel i testoval. Vzhledem k tomu, že jsem nepoužil žádné nadstandardní knihovny pro tvorbu GUI, měl by být projekt bez problému přenositelný na platformu MS Windows. Toto je významná vlastnost projektu, protože potenciální uživatelé systému jsou pravděpodobně uživatelé některé verze tohoto OS.

Pro psaní kódu jsem zvolil Gedit, což je základní textový editor pod Gnome. Nejedná se o žádné extrémně sofistikované prostředí jako je třeba Netbeans nebo Visual studio (je to přece jen textový editor). Ale pro vývoj projektu rozsahu mé práce je to nástroj naprosto dostačující. Velkou výhodou tohoto přístupu je možnost psát si vlastní makefile. V kombinaci s jednoduchými skripty v Shellu jsem tak měl k dispozici velice kvalitní a efektivní prostředí pro vývoj projektu.

Při psaní kódu jsem se snažil využívat výhradně anglické názvosloví pro názvy proměnných, komentáře i textové výpisy do terminálu. Tento přístup jsem zvolil z toho důvodu, aby byl můj zdrojový kód lépe čitelný pro případné zájemce ze zahraničí. Pro jistotu uvádím i své dva e-maily, kdyby se náhodou našel někdo, kdo by potřeboval konzultaci k některým nejasnostem, na které je možné při studii kódu narazit.

I když se jedná o rozsáhlejší projekt, probíhala implementace poměrně snadno a úspěšně. Jediné závažnější problémy nastaly při použití knihovny Gecode. Některé principy jejího fungování není totiž snadné pochopit. Je sice pravda, že uživatel má k dispozici kvalitní dokumentaci v podobě knihy Modeling with Gecode [12], ale jedná se o velmi náročný text, který je dostupný pouze v angličtině. Bohužel také zatím není k dispozici žádný tutorial nebo jednoduchý návod pro začátečníky. Při studiu Gecode mi však velmi pomohly konzultace s ostatními studenty FIT, kteří používali tuto knihovnu k řešení svých bakalářských projektů. Tyto konzultace vedl vedoucí mé práce Ing. Michal Fapšo a myslím, že byly velkým přínosem nejen pro mě, ale i pro mé kolegy. Přes počáteční problémy se knihovna Gecode ukázala jako velmi vhodný a efektivní nástroj pro řešení problémů spojených s omezujícími podmínkami.

5.1 Implementace jednotlivých modulů

5.1.1 Hlavní modul

Hlavní modul je reprezentován soubory `main.cpp` a `main.h`. Soubor `main.cpp` obsluhuje a řídí běh celé aplikace. Tuto správu má za úkol funkce `main`, což je typický znak nejen jazyka C, ale

i mnoha dalších programovacích jazyků. Další funkce obstarávají inicializaci struktur a pomocné konzolové výpisy.

Nejvýznamnějším hlavičkovým souborem celé aplikace je `main.h`. Obsahuje deklarace všech funkcí pro jednotlivé moduly, dále deklarace struktur a globálních proměnných, všechna používaná makra a vložení hlavičkových souborů jednotlivých knihoven.

Struktura reprezentující jednotlivé hlasy melodie se nazývá `TVoice`. Základem jsou dvě dynamická pole. První z nich je typu `int` a reprezentuje výšky jednotlivých tónů. Tyto hodnoty výšek korespondují s tím jak jsou noty reprezentovány v MIDI souborech. Druhé pole je typu `float` a reprezentuje délky not. Tato reprezentace vychází z časového údaje délky jednotlivých not. Základní hodnotou je 1, což odpovídá šestnáctinové notě (kratší notové délky se u varhanních předeher prakticky nevyskytují, nicméně je možné je teoreticky reprezentovat pomocí kladných reálných čísel menších než 1). Systém potom uvažuje následující notové délky:

- šestnáctinová,
- osminová triola,
- osminová,
- osminová s tečkou,
- čtvrťová,
- čtvrťová s tečkou,
- půlová,
- půlová s tečkou,
- celá nota.

Pomocí těchto notových délek jsem si vystačil při tvorbě všech potřebných předeher. Pokud by však bylo třeba rozšířit množinu notových délek o další hodnoty, nebyl by to problém. Nevýhodou reprezentace pomocí polí je pomalý sekvenční přístup k jednotlivým notám. Mohl jsem tedy zvolit nějaký sofistikovanější přístup k reprezentaci not (viz kapitola 3.2). Nespornou výhodou mého přístupu však je přímá vazba na vstup pro Gecode (zde je ideálním vstupem pole prvků typu int). Nakonec jsem se proto rozhodl pro reprezentaci not pomocí polí.

Předehra je potom reprezentována pomocí struktury `TOverture`. Tato struktura je jakýmsi kontejnerem not, který obsahuje pět struktur typu `TVoice`, které obsahují jednotlivé hlasy předehry (soprán, alt, tenor, bas), poslední hlas je potom cantus firmus, který není součástí výstupní předehry, ale je jakýmsi pomocným hlasem, ve kterém je uložena hlavní melodie předehry.

5.1.2 Modul pro čtení

Funkce pro čtení jsou definovány v souboru `read.h`. Význam a funkčnost tohoto modulu je specifikována v kapitole zabývající se implementací. Je však třeba doplnit, že se nejedná o plnohodnotný parser zdrojového kódu pro systém LilyPond. Předpokládá se jednohlasý vstup, který obsahuje pouze některé escape sekvence, které LilyPond nabízí. Návod k vytvoření korektního vstupu je součástí manuálu k aplikaci, který je součástí příloh této práce.

Následující příklad představuje ukázku takového vstupního souboru pro píseň Narodil se Kristus Pán. Tento vstup odpovídá notovému výstupu znázorněném na Obr. 2-1.

```
\score {
  \new Staff {
    \set Score.tempoHideNote = ##t
    \tempo 4 = 120  % set tempo to 120 quarters per minute
    \clef treble    % clef
    \key f \major   % scale
    \time 4/4       % time signature

    % tones of the melody
    f'4 f'4 a'4 b'4 c''4 c''4 c''2 b'4 c''4 d''2 c''1         \breathe
    c''4 c''4 bes'4 bes'4 a'4 a'4 g'2 a'4 bes'4 a'4 (g'4) f'1 \breathe
    a'4 g'4 a'4 bes'4 c''2 f'2 a'4 g'4 a'4 bes'4 c''2 f'2     \breathe
    f'2 d''2 c''4 bes'4 a'4 (g'4) f'1 \bar "|."

  }
}
```

Př. 5-1 Obsah vstupního souboru systému

5.1.3 Modul pro vytváření melodie

Modul pro vytváření melodie je implementován v podobě souboru `melody.h`. Tento modul pracuje vždy s jednou frází vstupní melodie. Pro následnou práci používá některou z těchto technik:

- opakování,
- inverze,
- račí postup.

Jak bylo vysvětleno v kapitole 2.3.1, existuje větší množství způsobů práce s tématem. Pro demonstraci funkčnosti systému jsem si však vystačil s těmito třemi. Systém je navrhnut tak, aby přidání dalších způsobů práce s tématem šlo provést jednoduše tím, že se vytvoří funkce, která implementuje daný přístup práce s tématem.

5.1.4 Modul pro harmonizaci

Modul pro harmonizaci jsem nakonec rozdělil do čtyř souborů. Průběh harmonizace je řízen pomocí funkcí implementovaných souborem `harm.h`. Tento soubor zajišťuje provedení mnoha pomocných funkcí, které jsou potřebné pro proces harmonizace. Soubory `hbas.h`, `halt.h` a `bten.h` potom slouží k postupnému vytváření jednotlivých doprovodných hlasů. V souladu s návrhem jsem pro tuto činnost využil knihovnu Gecode.

Pro harmonizaci jsem použil celkem třináct pravidel. Šest z nich ošetřuje obecná pravidla pro harmonizaci, zbylých sedm definuje prioritu jednotlivých harmonických funkcí. Vzhledem k tomu, že některá pravidla mohou být vzájemně v rozporu (v jednom hlase nelze například použít dvě harmonické funkce zároveň), bylo třeba zavést váhování pravidel. Váha každého pravidla je vyjádřena přirozeným číslem z intervalu <1.. 100>. Čím vyšší číslo, tím více se bude systém snažit, aby bylo dané pravidlo splněno. Gecode se potom snaží nalézt řešení, které má nejvyšší váhu. V následujících odstavcích jsem přiblížil význam jednotlivých pravidel pro harmonizaci z pohledu nauky o harmonii:

RULE_FIRST

Pokud je to možné, snaží se systém reprezentovat první akord pomocí tóniky[25]. Tónika dává uživateli poměrně dobrou představu o tonálním charakteru předehry.

RULE_ENDING

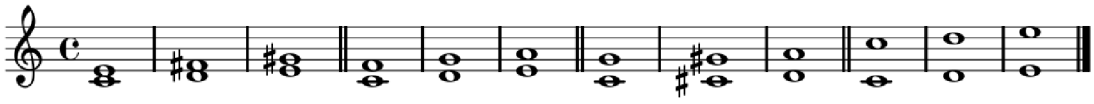
Závěr je harmonicky správné a hudebně přesvědčivé ukončení hudební věty. V technickém slova smyslu rozumíme v hudbě pod pojmem závěr spojení posledních dvou akordů v hudební větě [3]. Jedná se tedy o pravidlo pro vytváření závěrů, což je jedno z nejdůležitějších pravidel klasické harmonizace. Ve své nejjednodušší podobě vyjadřuje, že harmonizace by měla být zakončena spojem dominanta[26], tónika. To odpovídá závěru autentickému celému.

RULE_SCALE

Tónina je příslušnost tónového materiálu k určité stupnici. Mluvíme-li tedy o skladbě, neříkáme, že je ve stupnici G-dur, ale v tónině G-dur. To vyjadřuje, že skladatel použil při její tvorbě převážně tóny ze stupnice G-dur [16]. Tonální základ potom vyjadřuje jakousi soudržnost skladby. Pravidlo RULE_SCALE tedy vyjadřuje míru snahy systému používat doškální tóny.

RULE_PARALLEL

Toto pravidlo vyjadřuje omezení paralelních postupů jednotlivých hlasů v jednotlivých intervalech. Tyto postupy jsou pro mnohé intervaly zakázány. Paralelní postup je takový spoj, kdy hlasy postupují ve stejných intervalech. Pro bližší představu uvádím schematický obrázek (viz Obr. 5-1). Pravidlo povoluje v souladu s hudební teorií paralelní postupy pouze pro následující intervaly: tercie, kvarta, sexta.



Obr. 5-1 Paralelní intervaly (postupně paralelní tercie, kvarty, kvinty a oktávy)

RULE_INTERVALS

Toto pravidlo ošetřuje intervalovou vzdálenost mezi jednotlivými hlasy. Tím tak zabraňuje vzniku disonantních intervalů jako je například sekunda nebo septima.

RULE_MOVE

V případě tohoto pravidla se jedná o snahu, aby mezi jednotlivými tóny daného hlasu byl pohyb. Zakazuje tedy dva po sobě jdoucí stejné tóny v jednom hlase.

25 Tónika = doškální kvintakord vystavěný na prvním stupni daného modu.

26 Dominanta = doškální kvintakord vystavěný na pátém stupni daného modu.

RULE_TON, RULE_DOM, RULE_SUB, RULE_II, RULE_III, RULE_VI, RULE_VII

Aplikace těchto pravidel zajišťuje váhu, která je přidělena jednotlivým harmonickým funkcím daného modu[27]. Tím je zajištěna reprezentace významu jednotlivých funkcí. Tento význam je patrný v doprovodu téměř každé písně. Tyto doprovody se převážně opírají o hlavní harmonické funkce (tóniku, dominantu a subdominantu[28]).

Aplikaci všech těchto pravidel má uživatel možnost ovlivnit tím, že změní jejich váhu. To je možné provádět editací souboru `config`. Výchozí obsah tohoto souboru reprezentuje mnou zvolené optimum nastavení hodnot pro váhy těchto pravidel. Pro snadnější orientaci při editaci souboru, je tento doplněn o komentáře.

```
//////////////////////////////////////////////////////////////////////
// RULES FOR THE HARMONIZER
// - you can edit the number representing the particular rule
// - the number must be from interval <1..100>
//////////////////////////////////////////////////////////////////////
RULE_FIRST      50 // first chord should be T
RULE_ENDING    100 // harmonization ending by D-T
RULE_SCALE      75 // if possible use notes from the input scale
RULE_PARALLEL   75 // allowed parallel intervals are m3, v3, c4, v6
RULE_INTERVALS 100 // allowed intervals between voices are c1, m3, v3, c5
RULE_MOVE       10 // 2 neighbouring notes should be different
RULE_TON        15 // priority of T
RULE_DOM        12 // priority of D
RULE_SUB        10 // priority of S
RULE_2ND         2 // priority of 2ND
RULE_3RD         4 // priority of 3RD
RULE_6TH         4 // priority of 6TH
RULE_7TH         2 // priority of 7TH
```

Př. 5-2 Výchozí nastavení konfiguračního souboru

5.1.5 Modul pro vytváření předeher

Modul pro vytváření předeher je spolu s modulem pro harmonizaci a modulem pro vytváření vstupní melodie jádrem celé aplikace. Řídící soubor modulu pro vytváření polyfonních předeher je `overture.h`. Tento modul volá na základě vstupních parametrů vždy příslušné funkce jednoho ze dvou pomocných modulů `ohomo.h` a `opoly.h`. Kromě toho obsahuje ještě několik pomocných funkcí, které slouží k práci s jednotlivými hlasy předehry. Modul `ohomo.h` potom vytváří předehry v homofonním stylu, naproti tomu modul `opoly.h` implementuje funkce pro tvorbu předeher stylu polyfonního.

Mým původním záměrem bylo implementovat všechny styly předeher, které jsem představil v kapitole 2. Tento záměr se mi podařilo víceméně naplnit. Vynechal jsem pouze echo a fugato. Echo

27 Modus = synonymum pro stupnici v hudbě.

28 Subdominanta = doškální kvintakord vystavěný na čtvrtém stupni daného modu.

jsem vynechal kvůli tomu, že i bez něj mám již 6 stylů pro homofonní předehry. Fugato je naproti tomu natolik odlišným a náročným stylem, že by postačilo na téma některé jiné diplomové práce, která by se mohla zabývat například expozicí fugy.
Výsledná aplikace tedy umí vytvářet celkem 10 typů předeher různých stylů:

- duo (duo),
- duo s figurovaným basem (duo with figured bass),
- duo s figurovaným kantem firmem (duo with figured cantus firmus),
- rytmizovaná harmonizace (rhythmized harmonization),
- melodie v base (melody in bass),
- melodie v tenoru (melody in tenor),
- responsoriální přístup (responsorial),
- imitace (imitation),
- ostinato (ostinato),
- melodie na prodlevě (drone).

V závorkách jsou uvedeny anglické názvy příslušných stylů, protože tyto názvy jsou použity v programu.

5.1.6 Modul pro výstup

Problematiku vytvoření textového výstupu aplikace ošetřují funkce implementované v souboru `lily.h`. Pro snadnější orientaci v později vygenerovaných výstupech grafických výstupech je na začátku každého výstupního souboru kód, který generuje popisek k dané předehře. Tento popisek vychází z názvu vstupního souboru a stylu, který byl pro předehru použit.

5.1.7 Pomocné moduly

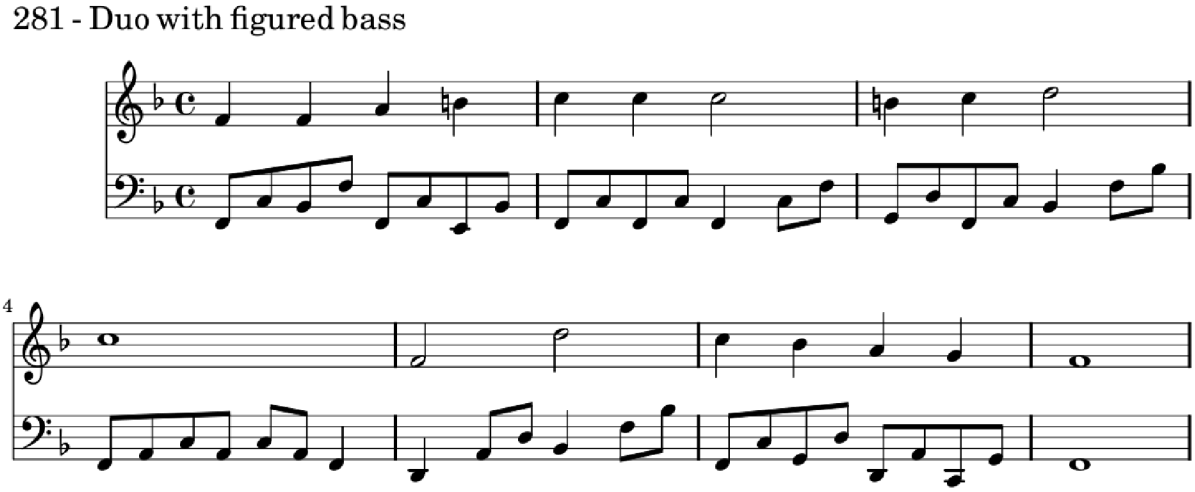
Pomocné moduly jsou tyto: `errors.h`, `memory.h`, `params.h` a `stats.h`. Tyto moduly jsem nazval pomocnými, protože neřeší přímo implementaci problematiky vytvoření varhanní předehry k dané vstupní melodii. Jsou však přesto nezbytné pro správné fungování celé aplikace.

Soubor `errors.h` má na starosti výpis chybových hlášení pokud nastane nějaký problém, po kterém je třeba aplikaci ukončit. Jedná se zejména o chybný vstup nebo o problémy spjaté s alokací paměti. Soubor `memory.h` implementuje funkce pro správu paměti, tedy její dynamickou alokaci a následné uvolnění. Soubor `params.h` zajišťuje správné zpracování vstupních parametrů programu a zpracování vstupního souboru `config`. Umožňuje rovněž výpis nápovědy do konzole. Posledním pomocným modulem je soubor `stats.h`, který počítá vybrané metriky kódu. Těmito metrikami jsou počet řádků a znaků zdrojového kódu v jednotlivých souborech.

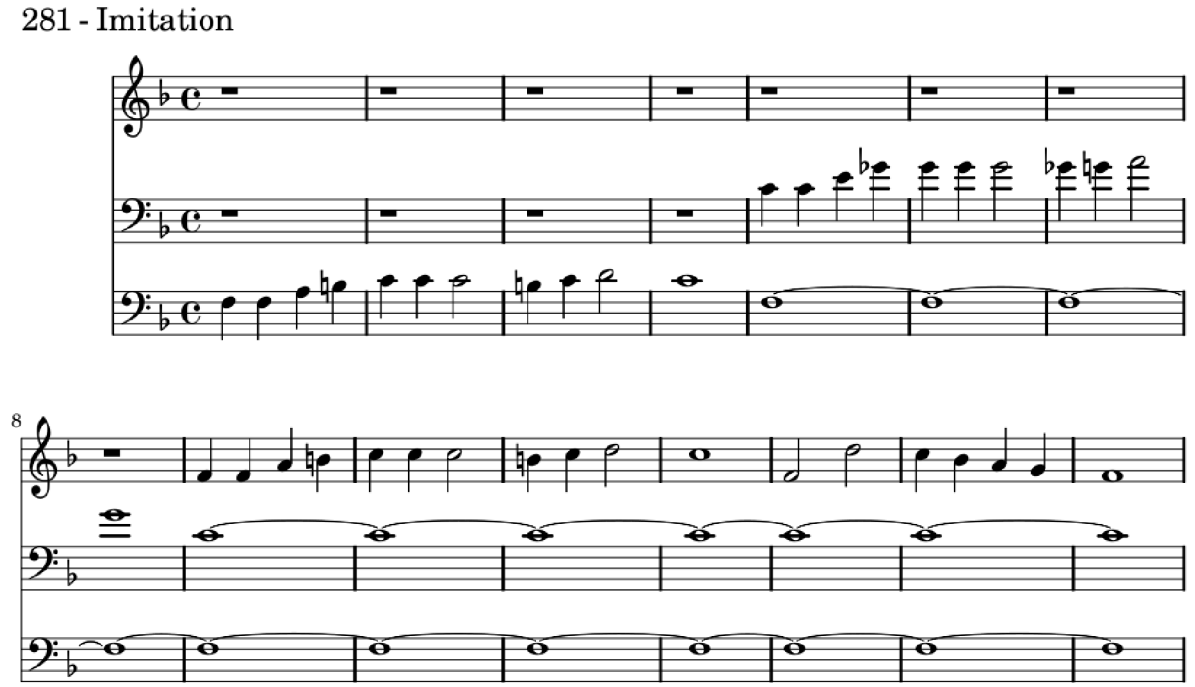
5.2 Ukázka výstupu aplikace

Příklady výstupů aplikace jsou součástí příloh této práce. Tyto výstupy se skládají z notových zápisů ve formátech pdf a ps, zvukových ukázek ve formátech mp3 a midi a zdrojového kódu pro LilyPond v textové podobě.

Přesto jsem se rozhodl zařadit pro ilustraci dvě notové ukázky i do hlavní textové části této práce. Obě tyto ukázky jsou předehrami k písni Narodil se Kristus Pán, na jejímž příkladu jsem rovněž vysvětloval problematiku varhanních předeher z teoretického hlediska.



Obr. 5-2 Ukázka notového zápisu, který je jedním z výstupů sytému



Obr. 5-3 Ukázka notového zápisu, který je jedním z výstupů sytému

6 Budoucí vývoj projektu

V této kapitole se pokusím nastínit, jak by mohl pokračovat vývoj tohoto projektu v budoucnosti. Vzhledem k tomu, že se jedná o poměrně rozsáhlou problematiku, je zde mnoho možností, jak tento program dále vyvíjet. Tato skutečnost byla zřejmá už od počátku vývoje, proto jsem se snažil program navrhnout tak, aby byl další vývoj možný.

Tato kapitola se skládá ze dvou částí. První z nich se zabývá zdokonalením vlastního systému, druhá se pokusí přiblížit možnost dalšího vývoje v souvislosti se spoluprací s Janáčkovou akademií múzických umění (dále jen JAMU).

6.1 Zdokonalení systému

V následujících kapitolách je nastíněna možnost, jak by se aplikace mohla dále vyvíjet a zdokonalovat. Je třeba říci, že je ještě na čem pracovat, protože možností k jejímu zdokonalení je opravdu mnoho. Podle mého názoru se jedná o dobrou vlastnost každé aplikace. Vždy je třeba mít možnost ji dále vyvíjet a zdokonalovat. V případě některých úprav se jedná o triviální záležitost. Jiná rozšíření jsou však poměrně složitá a náročná na provedení. Aplikaci jsem se však snažil navrhnout tak, aby všechna tato rozšíření byla možná.

6.1.1 Možnost MIDI vstupu

Pokud chce uživatel používat současnou verzi systému, musí jako vstup vytvořit zdrojový kód pro aplikaci LilyPond, který obsahuje melodii dané písně. To však zabere jistý čas, a proto se to může jevit uživateli jako zbytečně zdlouhavý a pracný úkol. Pro větší pohodlnost při práci se systémem by tedy bylo vhodné zavést a otestovat MIDI vstup. Uživatel by potom mohl zadávat melodii vstupní písně pohodlněji prostřednictvím kláves s MIDI rozhraním pro výstup.

Použití MIDI vstupu je teoreticky možné již v současnosti díky nástroji midi2ly. Tento přístup jsem však zatím netestoval, je tedy otázkou, jak přesně bude vypadat zdrojový kód získaný pomocí toho nástroje. Je vysoce pravděpodobné, že bude třeba upravit soubor `read.h`. Ve své podstatě se však jedná o jednoduchou úpravu.

6.1.2 Práce s tématem

Současný systém využívá pouze několika přístupů pro práci s tématem (opakování, inverze a račí postup). Tuto množinu přístupů by bylo vhodné rozšířit o další (dělení, krácení, rozšiřování apod.). Toto rozšíření je opět poměrně triviální záležitostí.

Ideální systém by však měl mít schopnost jednotlivé přístupy práce s tématem vhodným způsobem kombinovat. Pokud by měl systém tyto přístupy kombinovat nějakým sofistikovaným způsobem, jednalo by se už o poměrně netriviální rozšíření. Jeho případná implementace vyžaduje hlubší studium dané problematiky. Přesto si myslím, že by se jednalo o velmi přínosné rozšíření, které by systému otevřelo zcela nové možnosti. Systém je proto navržen a implementován tak, že s daným rozšířením počítá.

6.1.3 Rozšíření množiny pravidel pro harmonizaci

Další neméně zajímavou a přínosnou cestou, kterou by se mohl systém vyvíjet, je možnost zdokonalení harmonizeru. Ten sice v současné podobě funguje poměrně uspokojivě a navíc je jeho výstup ovlivnitelný uživatelem. Přesto je třeba říci, že ošetřuje jenom několik málo pravidel pro harmonizaci. Prostřednictvím vah případných nových pravidel by měl uživatel lepší kontrolu nad výstupem tohoto modulu. Výstup by tak mohl lépe odpovídat představám klasické harmonie. Navíc by bylo možné definovat si rozmanitější a zajímavější styly pro harmonizaci.

Takové rozšíření je z hlediska implementace poměrně nenáročné. Stačí totiž pouze přidat daná pravidla do kódu, který ošetřuje jejich definici pomocí Gecode. Trochu složitějším problémem by však bylo vymyslet takovou rozšiřující množinu pravidel. K tomuto řešení mám v úmyslu využít další konzultace s odborníky v této oblasti.

Ideální systém pro harmonizaci by pak šel ještě dál. Nabízel by totiž jednoduché intuitivní rozhraní pro definici vlastních pravidel (a ne pouze jejich vah). Zatím se mi však nepodařilo vytvořit návrh takové funkcionality. Problém je v tom, že systém by musel být příliš složitý na ovládání uživatelem.

6.1.4 Styly varhanních předeher

Aplikaci se již nyní daří poměrně úspěšně implementovat deset forem varhanních předeher. Tím však vývoj modulů implementujících tuto problematiku nekončí. Naopak na daném základě se dá dále stavět a ubírat se kupředu. Vývoj stylů se může ubírat dvěma způsoby, které se pokusím nastínit v následujících odstavcích.

První možností vývoje je zdokonalení současných stylů předeher. Příkladem takových rozšířením může být možnost výběru rytmické smyčky pro rytmizovanou harmonizaci. Jinou možností zdokonalení může být například výběr figury pro ostinato nebo lepší vedení figurovaného basu pro duo. Myslím, že takových drobných úprav je možné aplikovat velké množství. Výhodou takové rozšíření je rychlý a viditelný výsledek.

Druhou možností je zavedení nových stylů předeher. V kapitole 2 byl představen pouze výběr z nich. Ve skutečnosti je jich však daleko více. V poslední době jsem se rovněž na teoretické úrovni zabýval expozicí fugy, která je úzce spjata se stylem fugato. Je pravda, že by se jednalo o náročnou práci, výsledek by však mohl být velmi zajímavý.

6.2 Spolupráce s JAMU

S ostatními studenty jsme během tvorby svých prací navázali spolupráci s JAMU. Jednalo se zatím pouze o jednu schůzku několika studentů FIT s Danielem Dlouhým, Ph.D.

Daniel Dlouhý je český hudební skladatel a hráč na bicí nástroje. Narodil se roku 1965 v Brně, kde rovněž studoval na VUT (1983-1988), dále na JAMU obor Bicí nástroje (1988-1992), poté studoval kompozici (1993-1999) u Aloise Piňose a obor Teorie kompozice (1999-2002). V roce 1985 se stal členem brněnského souboru Art Inkognito. Z bicí sekce této formace roku 1990 spolu s Adamem Kubíčkem založil Středoevropský soubor bicích nástrojů DAMA DAMA, jehož je uměleckým vedoucím a dramaturgem. V současnosti působí jako interpret na bicí nástroje, hudební skladatel a učitel na JAMU [17].

Na schůzce s tímto hudebním skladatelem jsme diskutovali o tom, co se na naší fakultě řeší v rámci diplomových a bakalářských projektů spjatých s hudbou. Je pravda, že dosavadní vývoj mého projektu nebyl tímto setkáním příliš ovlivněn. Myslím si však, že do budoucna má spolupráce s JAMU velký potenciál. Připomínky a návrhy profesionálních hudebníků mohou totiž pomoci zkvalitnit naše práce. Na podzimní semestr je předběžně naplánovaná prezentace závěrečných prací studentů FIT studentům kompozice na JAMU v rámci jejich výuky.

7 Závěr

Úspěšně jsem navrhl a vytvořil program, který je schopen vytvářet varhanní předehry k církevním písním. Výstup programu je ovlivnitelný uživatelem. Ten může vybrat, se kterou částí vstupní melodie se bude dále pracovat, nastavit váhu pravidel pro harmonizaci a zvolit styl, ve kterém bude daná předehra vytvořena.

Snažil jsem se, aby nebylo náročné s programem pracovat, což se mi z velké části podařilo. V současné verzi však zatím chybí možnost MIDI vstupu z kláves. Tato možnost má potenciál značně zvýšit uživatelský komfort. Výhodou sytému je možnost notového i zvukového výstupu. Obojího jsem dosáhl díky aplikaci LilyPond.

Při návrhu aplikace jsem vycházel z hudební teorie. Předpokládal jsem tedy využití principů základních metod práce s tématem a stylů pro varhanní předehry. Důležitou inspirací mi potom byl systém Strasheela, který používá k řešení automatické kompozice metodiku CSP. Tento přístup jsem využil v rámci modulů, které mají na starost vytvoření harmonizace melodie předehry.

Pro implementaci jsem zvolil programovací jazyk C++ s knihovnou Gecode pro řešení CSP. Oba tyto prostředky se ukázaly být velmi efektivními nástroji pro vývoj mého systému. Jedinou věcí, která komplikovala vývoj aplikace, byla náročnost práce s knihovnou Gecode. Nakonec se však ukázalo, že bylo vhodné tento nástroj použít. Během implementace jsem zachoval původní návrh celého systému. Při psaní kódu jsem používal výhradně angličtinu, aby bylo možné projekt využít k mezinárodní spolupráci.

Systém jsem implementoval a testoval v prostředí operačního systému Ubuntu 9.10 Karmic Koala, kladl jsem však důraz na platformní nezávislost. Myslím si, že v dnešní době by totiž byla chyba omezovat se na jednu platformu. Navíc je velmi pravděpodobné, že případní uživatelé systému budou zvyklí na práci s OS Microsoft Windows. Během testování se potvrdilo, že program splňuje zadání projektu a je schopen vytvářet jednoduché varhanní předehry.

7.1 Přínos práce

Podařilo se mi naplnit cíle, které jsem si stanovil v kapitole (1.3). Vytvořil jsem tedy prakticky použitelného pomocníka pro varhaníka, který je schopen komponovat jednoduché varhanní předehry k církevním písním. Příklad notového výstupu programu je uveden v kapitole 5.2. Větší množství notových i zvukových výstupů je potom ve složce `examples` na přiloženém CD.

V rámci testování jsem výstup programu použil i prakticky během bohoslužeb, kdy jsem si nechal vytvořit varhanní předehry k některým písním. Tento experiment byl úspěšný. Program je tedy v zásadě použitelný i pro ostatní varhaníky při vykonávání jejich praxe. V současnosti používá většina uživatelů některou distribuci operačního systémů MS Windows. Proto je třeba se ještě zaměřit na zprovoznění aplikace na těchto platformách. To by však neměl být velký problém, protože se jedná o konzolovou aplikaci.

Dalším přínosem mé práce může být navázaná spolupráce s oborem kompozice na JAMU. Myslím si, že tato mezioborová spolupráce najde uplatnění i při dalším výzkumu systémů pro automatickou kompozici na naší fakultě.

7.2 Budoucí vývoj projektu

Jelikož se můj projekt zabývá poměrně rozsáhlou problematikou, rozhodl jsem se budoucímu vývoji věnovat celou šestou kapitolu. V ní jsem nastínil, jakou cestou vývoje by se celý projekt mohl dále ubírat. Tuto svou vizi bych rád nejprve zkonzultoval s případnými uživateli systému a studenty kompozice na JAMU.

Myslím, že připomínky obou těchto skupin uživatelů mohou být velmi věcné a praktické. Rád bych potom do systému zavedl většinu rozšíření, která jsem navrhl v kapitole 6.1.

Literatura

[1] Maňák, O.: *Harmonizace melodie*, bakalářská práce, Brno, FIT VUT v Brně, 2008 <http://www.fit.vutbr.cz/~ifapso/doc/music/bp_dp/Harmonizacia_-_Ondrej_Manak07.pdf>

[2] Silná, I.: *Nauka o hudebních formách*. Konzervatoř P. J. Vejvanovského Kroměříž 1999

[3] Silná, I.: *Nauka o harmonii*. Konzervatoř P. J. Vejvanovského Kroměříž, 1992

[4] Silná, I.: *Nauka o kontrapunktu*. Konzervatoř P. J. Vejvanovského Kroměříž, 1998

[5] Bělský, V.: *Nauka o varhanách*. Editio Bärenheiter Praha, 2000

[6] Järveläinen, H.: *Algorithmic Musical Composition*, Helsinki University of Technology, 2000 <http://www.tml.tkk.fi/Studies/Tik-111.080/2000/papers/hanna/alco.pdf>

[7] Angelov, M.: *Hudební improvizace*, bakalářská práce, Brno, FIT VUT v Brně, 2009 <http://www.fit.vutbr.cz/~ifapso/doc/music/bp_dp/Improvizacia_-_Michael_Angelov08.pdf>

[8] Vlasák, J.: *Harmonizace melodie*, bakalářská práce, Brno, FIT VUT v Brně, 2009 <http://www.fit.vutbr.cz/~ifapso/doc/music/bp_dp/Harmonizacia_-_Jaroslav_Vlasak08.pdf>

[9] Torsten, A.: *Music Representation*, Interdisciplinary Centre for Computer Music Research (ICCMR) University of Plymouth , 2007 <http://cmr.soc.plymouth.ac.uk/tanders/teaching/TorstenAnders-AINT503-e2.pdf>

[10] Barták, R.: *Expert Systems based on Constraints*. Katedra teoretické informatiky Matamaticko-fyzikální fakulty UK, Praha 1997 <http://kti.mff.cuni.cz/~bartak/downloads/Disertace.pdf>

[11] Zbořil, F., Zbořil, F.: *Základy umělé inteligence*. FIT VUT v Brně, 2006

[12] Schulte, Ch., Tack, G., Lagerkvist, M.: *Modelling and Programming with Gecode*. 2009 <http://www.gecode.org/doc-latest/MPG.pdf>

[13] Anders Torsen biography <http://cmr.soc.plymouth.ac.uk/tanders/index.htm>

[14] Strasheela for musicians <http://strasheela.sourceforge.net/strasheela/doc/index.html>

[15] Hutárek, J., Švéda, P.: *Software pro zápis not*, FIT VUT v Brně, 2009

[16] Zenkl, L.: *ABC hudební nauky*. Praha, Editio Bärenreiter, 2003

[17] Daniel Dlouhý – Český hudební slovník <http://www.ceskyhudebnislovnik.cz>

Seznam příloh

Příloha 1. Slovníček pojmů

Vzhledem k tomu, že téma této práce je úzce spjato s hudební teorií a naukou o varhanách, není možno vyhnout se termínům, které se v těchto oborech používají. Tyto pojmy jsou vysvětleny na následujících řádcích. Pro jejich vysvětlení jsem často použil citace z uvedené odborné literatury:

1. Hudební téma = delší závažná myšlenka, která slouží jako kompoziční základ skladby [2].
2. Harmonizace = proces vytváření doprovodných hlasů na základě akordického hudebního myšlení [3].
3. Hudební forma = celkové rozvržení skladby a uspořádání jejích jednotlivých částí včetně hudebních myšlenek [2].
4. Varhanní rejstřík = sada píšťal stejné síly, barvy a charakteru zvuku [5].
5. Manuál = varhanní klaviatura pro ruce, varhany mají obvykle dvě nebo i více těchto klaviatur [5].
6. Žaluzie – v žaluziové skříni stojí obvykle rejstříky celého manuálu, jedná se o dřevěnou skříň, jejíž přední stěna je žaluziově rozdělena, otevíráním a zavíráním žaluzie se zvuk zesiluje a zeslabuje, zavřená žaluzie působí dojmem vzdálenějšího zvuku [5].
7. Josef Ferdinand Norbert Seger (1716-1782) byl český hudební skladatel, houslista a varhaník barokního období.
8. Motiv = nejmenší nedělitelný prvek ve skladbě, který má ještě určitý obsah a význam [2].
9. Opakování = nejjednodušší způsob práce s tématem, při němž není změněn ani rytmus ani melodie motivu [2].
10. Obměňování = menší nebo větší obměna melodie (změna velikostí intervalů), rytmu (změna délek rytmických hodnot), melodie i rytmu současně [2].
11. Ozdobování = motiv je obohacen o různé melodické tóny drobných hodnot, takže se jeho délka nemění [2].
12. Dělení = z motivu se vyjme jen určitá část a s ní se pracuje dále samostatně. Používá se zejména u delších motivů nebo témat [2].
13. Krácení (zkrácení) – vypouštění některých tónů nebo zkracování jejich délek. Celková délka motivu se tedy zkracuje [2].
14. Rozšíření (prodloužení) – vzniká přidáváním několika tónů nebo prodlužováním délek již existujících tónů. Celková délka motivu se tedy prodlužuje [2].
15. Převrat (inverze, obrat, protipohyb) = stoupající intervaly motivu nebo tématu se mění na klesající a naopak. Převrat může být buď přísný (intervaly jsou zachovány ve své velikosti) nebo volný (velikost jednotlivých intervalů se mění, opačný směr pohybu melodické linky však zůstává zachován) [2].
16. Račí postup = uvádění melodie, rytmu nebo obojího pozpátku [2].
17. Figurace hlasů = výzdoba hlasů melodickými tóny tak, aby se jednotlivé hlasy rytmicky doplňovaly [3].
18. Cantus firmus = obecně hlas, který obsahuje základní téma skladby (v našem případě se jedná o téma předehry).
19. Pedál = varhanní klaviatura pro nohy, která je obvykle položena v nižší poloze než manuál [2].

20. Pleno = typický silný varhanní zvuk, u menších varhan je tímto pojmem myšleno použití všech rejstříků.
21. Ostinátní figura = vícekráte se opakující krátký melodicko-rytmický útvar s menší závažností než motiv, v ostatních hlasech se přitom mění harmonie [3] .
22. Prodleva = delší dobu znějící tón nebo tóny, zatímco v ostatních hlasech se mění harmonie [3].
23. Doškální tón = tón, který se nachází v dané stupnici.
24. Digital audio workstation (DAW) = kompletní řešení pro produkci hudby a zvuku pomocí počítače. Mají potenciál nahradit i celé hardwarové vybavení hudebního studia [15].
25. Tónika = doškální kvintakord vystavěný na prvním stupni daného modu.
26. Dominanta = doškální kvintakord vystavěný na pátém stupni daného modu.
27. Modus = synonymum pro stupnici v hudbě.
28. Subdominanta = doškální kvintakord vystavěný na čtvrtém stupni daného modu.

Příloha 2. Obsah přiloženého CD

Součásti diplomové práce v elektronické podobě:

- text práce ve formátu pdf,
- text práce ve formátu odt,
- manuál k programu,
- zdrojové soubory programu,
- příklady vstupů a výstupů programu.

Příloha 3. Manuál k programu

Vytvoření vstupního souboru:

Šablonou pro vstup je komentovaný soubor `in_cz.ly` ve složce `/input`. Editací tohoto souboru je možno snadno vytvořit vstup pro aplikaci. Verze šablony s komentáři v anglickém jazyce nese název `in_en.ly`.

Editace souboru config:

Editací souboru `config` lze dosáhnout změny vah pravidel pro harmonizaci. Program potom harmonizuje jiným způsobem. Váhy vyjadřují snahu programu dodržet daná pravidla. Jsou určena přirozenými čísly v rozmezí 1 až 100. Význam jednotlivých pravidel je následující:

- RULE_FIRST: první akord by měl být tónika,
- RULE_ENDING: zakončení harmonizace spojem dominanta-tónika,
- RULE_SCALE: používání not z dané stupnice,
- RULE_PARALLEL: zákaz paralelních intervalů (kromě 3, 4, 6),
- RULE_INTERVALS: povolené intervaly mezi jednotlivými hlasy (1, 3, 4, 5, 6),
- RULE_MOVE: pohyb v base,
- RULE_TON – RULE_7TH: priority jednotlivých harmonických funkcí.

Spuštění programu:

Program lze spustit s následujícími parametry:

- -h: vytiskne nápovědu a ukončí program,
- -c: vytiskne statistiky kódu a ukončí program,
- -d: nastaví výchozí hodnotu souboru `config`,
- -i jmeno_souboru: nastaví vstupní soubor (ze složky `/input`),
- -o jmeno_souboru: nastaví výstupní soubor (ze složky `/output`),
- -e hodnota (40... 200): nastaví tempo předehry,
- -t hodnota (1... 3): vybere způsob práce s tématem:
    1: opakování,
    2: inverze,
    3: račí postup.
- -p hodnota (1... počet frází): vybere frázi melodie pro další zpracování,
- -s hodnota (1... 10): nastaví styl předehry:
    1: duo (duo),
    2: duo s figurovaným basem (duo with figured bass),
    3: duo s figurovaným kantem firmem (duo with figured cantus firmus),
    4: rytmizovaná harmonizace (rhythmized harmonization),
    5: melodie v base (melody in bass),
    6: melodie v tenoru (melody in tenor),
    7: responsoriální přístup (responsorial),
    8: imitace (imitation),

9: ostinato (ostinato),
10: melodie na prodlevě (drone).

Všechny parametry jsou nepovinné a na jejich pořadí nezáleží. Pokud není některý z nich zadán, je nastaven na implicitní hodnotu.

Program lze spouštět i pomocí jednoduchého skriptu `dip.sh`. Tento skript spustí program s danými parametry. Výhodou tohoto přístupu je možnost generování více předeher najednou. To lze udělat nastavením právě jednoho jednoho z parametrů 2, 3 nebo 4 na hodnotu nula (program je potom spuštěn se všemi možnými hodnotami tohoto parametru). Parametry skriptu `dip.sh`:

- 1. parametr: jméno vstupního souboru,
- 2. parametr: styl pro práci s tématem (0... 3),
- 3. parametr: fráze pro další zpracování (0... počet frází),
- 4. parametr: výběr stylu předehry (0... 10).

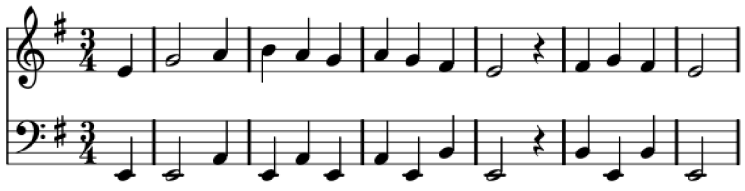
Vytvoření zvukového a notového výstupu:

Výstupem programu je zdrojový kód pro aplikaci LilyPond, který je uložený ve složce `/output`. Díky LilyPondu lze z tohoto souboru snadno vytvořit notový a zvukový výstup. Je tak třeba učinit pomocí následujícího příkazu: `lilypond jmeno_souboru`. Tímto způsobem jsou vytvořeny dva soubory s notovým výstupem (`jmeno_souboru.pdf` a `jmeno_souboru.ps`) a jeden soubor s výstupem notovým (`jmeno_souboru.midi`).

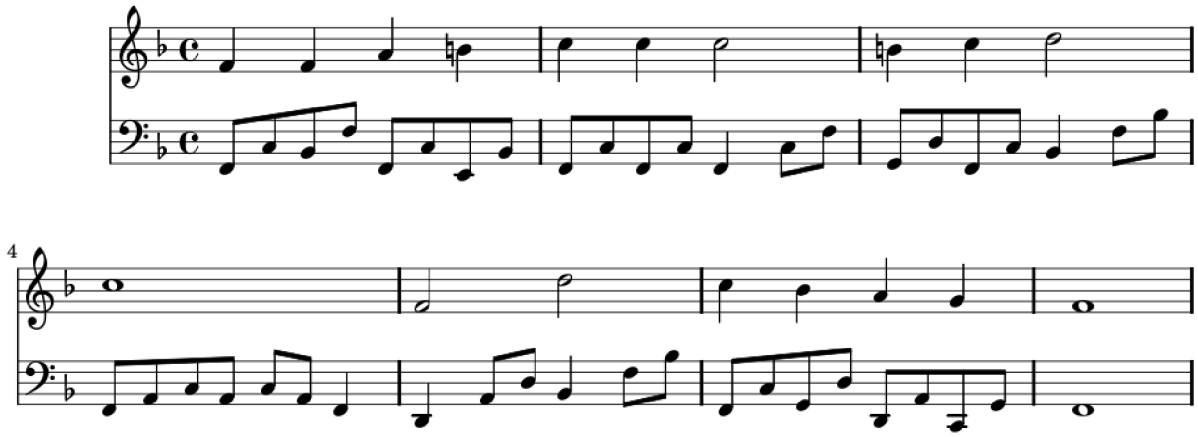
Jinou možností je využití jednoduchého připraveného skriptu `lily.sh`, který spustí LilyPond pro všechny jeho zdrojové soubory ve složce `/output`.

Příloha 4. Ukázky výstupů aplikace

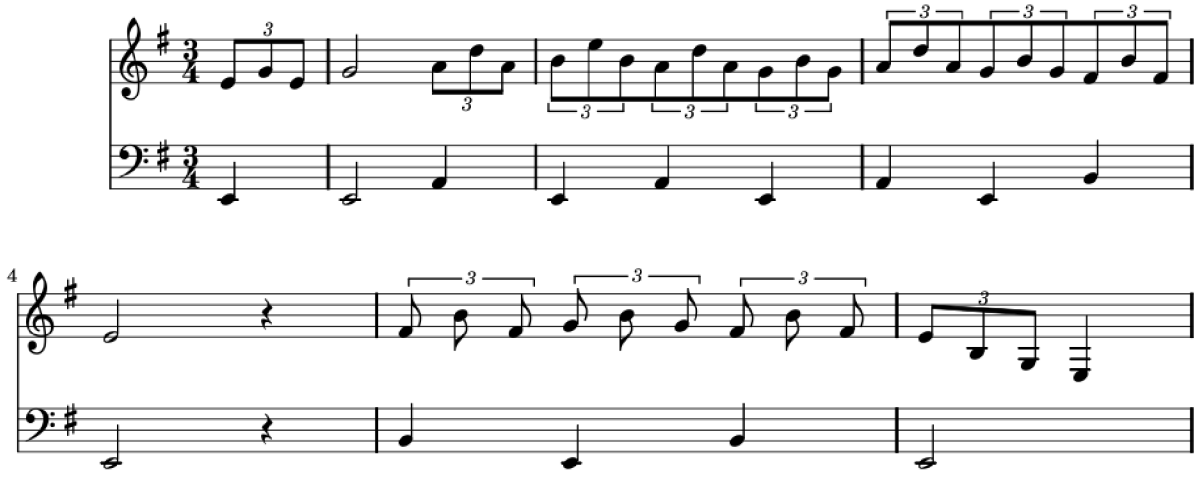
663 - Duo



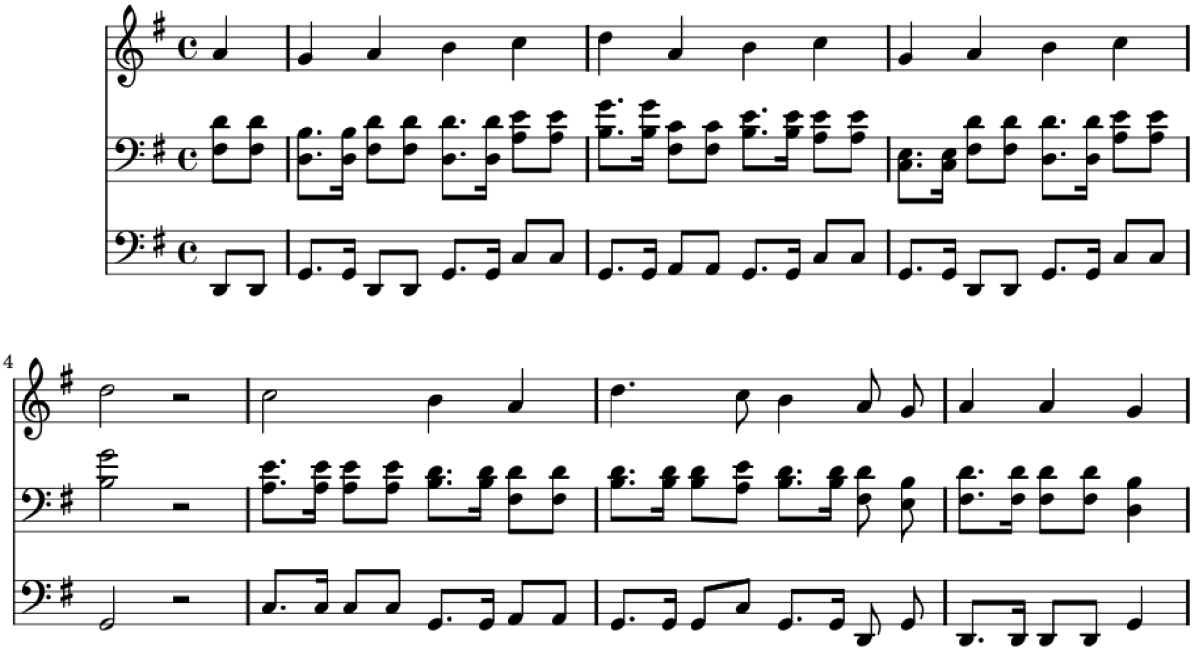
281 - Duo with figured bass



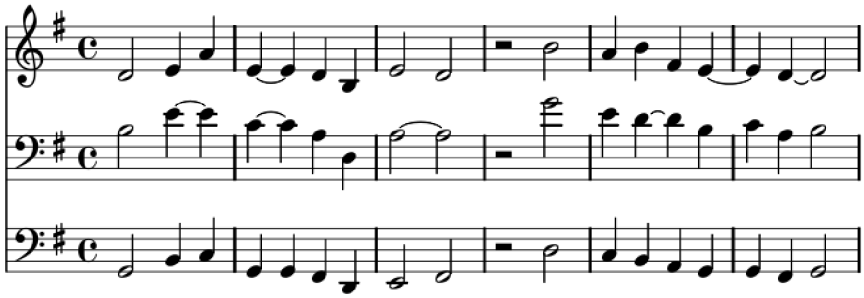
663 - Duo with figured cantus firmus



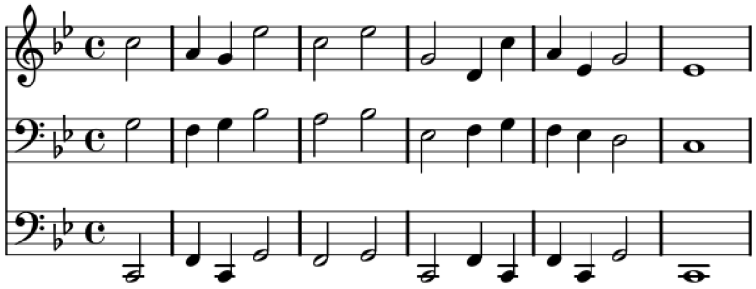
406 - Rhythmized harmonization



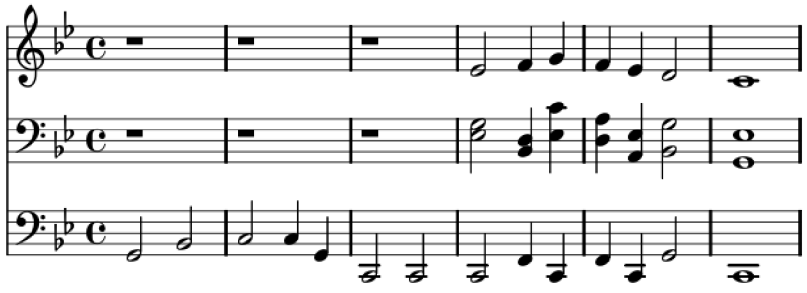
66 - Melody in bass



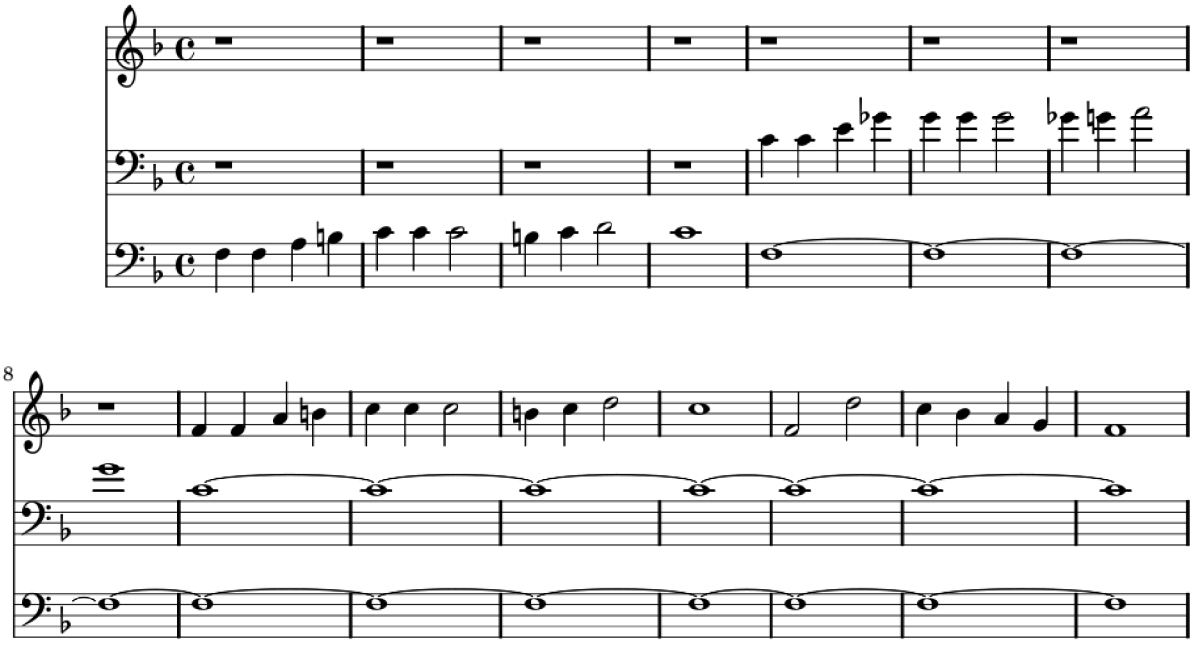
107 - Melody in tenor



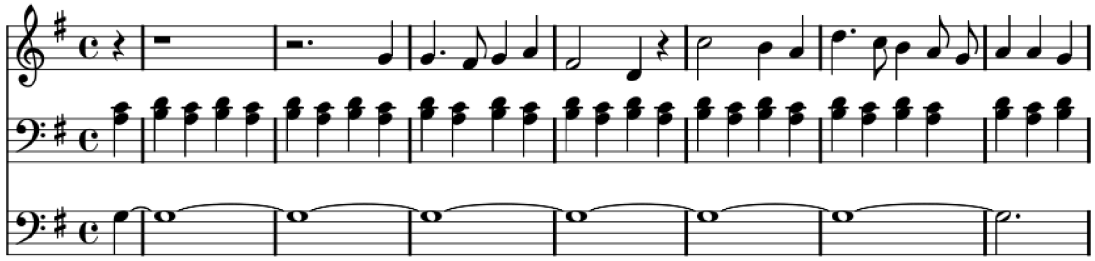
107 - Responsorial



281 - Imitation



406 - Ostinato



66 - Drone