天津大学

逻辑与形式化方法实验报告

音乐管理系统

2021 年 6 月 30 日

学院: 智能与计算学部

班级: 网络安全一班

姓名: 谢远峰

学号: 3019244283

目录

1 需求说明

本次作业设计音乐管理系统，具体事务和用户需求如下

1. 每个用户可以拥有多个歌单
2. 用户可以添加或删除一个或多个歌单
3. 用户可以在歌单内部添加或删除一个或多个歌曲
4. 用户可以通过音乐的作者或者主题查询音乐的具体信息
5. 用户可以查看当前播放音乐的具体信息
6. 用户可以暂停音乐的播放
7. 用户可以切换下一首音乐播放

同时，系统还需满足如下要求：

- 歌单内的歌曲要么未播放，要么正在播放
- 不存在某个歌曲既处于播放状态，又处于未播放状态
- 用户可以创建的歌单数量具有上限
- 用户单个歌单内可以包含的歌曲数量具有上限
- 用户在当前时刻只能播放一首音乐

2 形式规格说明

2.1 类型

[PERSON,LIST,MUSIC,AUTHOR,TITLE,SUBJECT]

PERSON 是所有可能的人的集合。LIST 是所有歌单的集合。MUSIC 是歌单内所有音乐的集合。AUTHOR 是所有可能的作者的集合。TITLE 是所有可能音乐名称的的集合。SUBJECT 是所有可能的主题的集合。

Repert::='*Success*' | '*Unknown_User*' | '*Too_Many_Playlist*' | '*Too_Many_Music*' | '*Music_Not_In*' | '*Music_Is_Playing*' | '*Music_Is_Playing*' | '*Playlist_In_Account*' | *Playlist_Not_In*'

添加特定的类型 REPORT，它是所有必要信息的集合，可由以上枚举类型来进行定义

2.2 常量

| $Playlist\_Limit, Music\_Limit : \mathbb{N}$

Playlist_Limit 表示用户最多可以创建的歌单数量，Music_Limit 表示用户在单个歌单内可以最多添加的歌曲数量

2.3 状态模式

USER

$user : \mathbb{P}\, PERSON$

$manager : \mathbb{P}\, PERSON$

$user \cap manager = \emptyset$

图 2.3.1: 用户状态模式 USER

模式 USER 定义了用户状态, 其中 user 表示普通用户,manager 表示管理员用户, 规定两种用户无交集

STORE

$list\_title : TITLE$

图 2.3.2: 模式 STORE: 定义歌单的基本信息

SONG

$title : TITLE$

$author : \mathbb{F}\ AUTHOR$

$subjects : \mathbb{F}\ SUBJECT$

图 2.3.3: 模式 SONG: 定义歌曲的基本信息

DB

$account : \mathbb{P}\ LIST$

$playlist : \mathbb{P}\ MUSIC$

$unplay : \mathbb{P}\ MUSIC$

$playing : \mathbb{P}\ MUSIC \pfun PERSON$

$last\_playing : \mathbb{P}\ MUSIC \pfun PERSON$

$music\_info : MUSIC \pfun SONG$

$list\_info : LIST \pfun STORE$

$(\mathrm{dom}\ playing) \cup unplay = playlist$

$(\mathrm{dom}\ playing) \cap unplay = \emptyset$

$\forall p : PERSON \bullet \#(playing \rhd \{p\}) \subseteq \{0, 1\}$

$playing \subseteq last\_playing$

$\mathrm{dom}\ music\_info = playlist$

$\mathrm{dom}\ list\_info = account$

图 2.3.4: 数据库状态模式 DB: 描述歌单和播放信息的数据库状态

LIB

USER

DB

$\mathrm{ran}\ playing \subseteq user$

图 2.3.5: 数据库状态模式 DB: 描述歌单和播放信息的数据库状态

$\Delta SONG \mathrel{\widehat=} SONG \land SONG'$

$\Delta USER \mathrel{\widehat=} USER \land USER'$

$\Delta DB \mathrel{\widehat=} DB \land DB'$

$\Delta LIB \mathrel{\widehat=} LIB \land LIB'$

$\Xi SONG \mathrel{\widehat=} \Delta SONG \mid title' = title \land author' = author \land subjects' = subjects$

$\Xi USER \mathrel{\widehat=} \Delta SONG \mid user' = user \land manager' = manager$

$\Xi DB \mathrel{\widehat=} \Delta DB \mid playlist' = playlist \land unplay' = unplay \land$
$playing' = playing \land last\_playing' = last\_playing \land music\_info' = music\_info \land$
$account' = account \land list\_info' = list\_info$

$\Xi LIB \mathrel{\widehat=} \Delta LIB \mid \theta USER' = \theta USER \land \theta DB' = \theta DB$

图 2.3.6: 状态模式定义:$\Delta SONG\ \Delta USER\ \Delta DB\ \Delta LIB\ \Xi SONG\ \Xi USER\ \Xi DB\ \Xi LIB$

2.4 状态模式初始化

系统初始化时，管理系统中使用用户，也无歌单和歌曲，但至少存在一个管理人员。由于需求中并未要求增加管理人员和增加读者等“用户控制系统”的事务，故假定管理人员集合和用户人员集合在初始化时已经设置好。可以假定管理员集合为 {happytraveller_alone}, 普通用户集合为 {traveller1、traveller2}

InitUSER

$$USER'$$

$$user' \neq \emptyset$$
$$manager' \neq \emptyset$$

图 2.4.1: 用户集群初始化

InitDB

$$DB'$$

$$account' = \emptyset$$
$$playlist' = \emptyset$$
$$music\_info = \emptyset$$
$$unplay' = \emptyset$$
$$playing' = \emptyset$$
$$last\_playing' = \emptyset$$
$$list\_info' = \emptyset$$

图 2.4.2: 数据库初始化

InitLIB

$$LIB'$$

$$InitUSER$$
$$InitDB$$

图 2.4.3: 组合模式初始化

显然 InitUSER 初始化状态存在，故证明 InitDB 及 InitLIB 的初始化定理
模式 InitDB 初始化定理可描述为 $\vdash \exists DB' \bullet InitDB$
展开该定理, 可得下式（1）

$$\begin{aligned}
\vdash \exists\, & playlist', unplay' : \mathbb{P}\, MUSIC; \\
& account' : LIST, list_info : LIST \pfun STORE; \\
& playing', last_playing : MUSIC \pfun PERSON; \\
& music_info : MUSIC \pfun SONG \\
& (\mathrm{dom}\ playing) \cup unplay = playlist \\
& (\mathrm{dom}\ playing) \cap unplay = \emptyset \\
& \forall p : PERSON \bullet \#(playing \rhd \{p\}) \subseteq \{0, 1\} \\
& playing \subseteq last_playing \\
& \mathrm{dom}\, music_info = playlist \bullet \\
& playlist' = \emptyset \land music_info = \emptyset \land unplay' = \emptyset \land \\
& playing' = \emptyset \land last_playing' = \emptyset
\end{aligned}$$

消除“|”, 并应用点规则，可得下式 (2)：

$$\begin{aligned}
\vdash\ & \emptyset \in \mathbb{P}\, MUSIC \land \\
& \emptyset \in \mathbb{P}\, LIST \land \emptyset \in \mathbb{P}\, MUSIC \land \\
& \emptyset \in \mathbb{P}\, LIST \pfun STORE \\
& \emptyset \in \mathbb{N} \pfun MUSIC \\
& \emptyset \in \mathbb{P}\, MUSIC \pfun PERSON \land \\
& \emptyset \in \mathbb{P}\, MUSIC \pfun PERSON \land \\
& \emptyset \in \mathbb{P}\, MUSIC \pfun SONG \\
& \emptyset \mapsto \emptyset \\
& (\mathrm{dom}\, \emptyset) \cup \emptyset = \emptyset \land (\mathrm{dom}\, \emptyset) \cap \emptyset = \emptyset \\
& \forall p : PERSON \bullet \#(\emptyset \rhd \{p\}) \subseteq \{0, 1\} \\
& \emptyset \subseteq \emptyset \land (\mathrm{dom}\, \emptyset) = \emptyset \land \mathrm{dom}\, \emptyset = \emptyset
\end{aligned}$$

子目标：

$(01) \vdash \emptyset \in \mathbb{P}\, MUSIC$

$(02) \vdash \emptyset \in \mathbb{P}\, MUSIC$

$(03) \vdash \emptyset \in \mathbb{P}\, MUSIC \pfun PERSON$

$(04) \vdash \emptyset \in \mathbb{P}\, MUSIC \pfun PERSON$

$(05) \vdash \emptyset \in \mathbb{P}\, MUSIC \pfun SONG$

$(06) \vdash (\mathrm{dom}\, \emptyset) \cup \emptyset = \emptyset$

$(07) \vdash (\mathrm{dom}\, \emptyset) \cap \emptyset = \emptyset$

$(08) \vdash \forall p : PERSON \bullet \#(\emptyset \rhd \{p\}) \subseteq \{0, 1\}$

$(09) \vdash \emptyset \subseteq \emptyset$

$(10) \vdash \mathrm{dom}\, \emptyset = \emptyset$

子目标 (1)(2) 可通过定理 12 证明，子目标 (3)(4)(5) 利用定理 14 证明。(6)(7) 利用定律 16 将 $\mathrm{dom}\, \emptyset$ 重写为 $\emptyset$，利用应用 L19 和 L25 可得到证明。(9) 利用定律 4 证明，(10) 由定律 6 证明，进行 (8) 的证明。

使用定律 L27, 重写为 $\forall P : PERSON \bullet \# \emptyset \leq \{0, 1\}$ 根据定理 L28，$\#\emptyset = 0$, 有 $\forall P : PERSON \bullet 0 \leq \{0, 1\}$

由数学定理可知：$\forall P : PERSON \bullet true$ 得证。

InitDB 初始化定理证毕。

模式 InitLIB 初始化定理可描述为 $\vdash \exists LIB' \bullet InitLIB$

由于 InitUSER 和 InitDB 相互独立且成立，结合点规则，可知 InitLIB 初始化状态存在

2.5 操作模式

用户需求事务可分为两种情况：管理人员事务与普通用户事务

2.5.1 管理人员事务

管理人员事务包括音乐管理事务，查询事务

所用管理人员事务包含以下特点：在不影响用户系统的前提下，需要管理人员的名称，引入模式 manager_trans,LIB 状态模式不受到改变。

manager_trans

ΔLIB

$id? : PERSON$

$id? \in manager$

$\theta USER' = \theta USER$

图 2.5.1: 管理人员事务模式 manager_trans

（1）歌单管理事务

歌单管理事务包括歌单的添加，删除，与歌单名称相关，引入管理事务 account_manage。

account_manage

$account\_manage$

$account? : LIST$

$playlist' = playing$

图 2.5.2: 管理人员歌单管理事务模式 account_manage

歌单删除事务需给确保用户包含歌单 playlist？, 需要修改 account' 和 list_info 的记录

playlist_remove

$account\_manage$

$playlist? \in account$

$account' = account \setminus (playlist?)$

$list\_info' = \{playlist\} \ndres list\_info$

图 2.5.3: 管理人员歌单管理删除事务模式 playlist_remove

歌单添加事务需给定歌单的名称 playlist？, 并提供歌单的名称。

playlist_add

$$
\begin{aligned}
&account\_manage \\
&playlist\_tilte? : TITLE \\
&\hline
&playlist? \notin account \\
&account' = account \cup (playlist?) \\
&list\_info' = list\_info \oplus \{playlist \mapsto B\} \\
&\quad where\ B : STORE \mid B.playlist\_tilte = playlist\_tilte?
\end{aligned}
$$

图 2.5.4: 管理人员歌单管理添加事务模式 playlist_add

（2）音乐管理事务

音乐管理事务包括歌单内音乐的添加，删除，且不影响用户歌曲播放，引入音乐管理事务 music_manage。

music_manage

$$
\begin{aligned}
&music\_manage \\
&music? : MUSIC \\
&\hline
&playing' = playing
\end{aligned}
$$

图 2.5.5: 管理人员歌曲管理事务模式 music_manage

歌曲删除事务需给确保用户包含歌曲 music？, 需要修改 unplay' 和 music_info 的记录

music_remove

$$
\begin{aligned}
&music\_manage \\
&\hline
&music? \in playlist \\
&playlist' = playlist \setminus (music?) \\
&unplay' = unplay \setminus (music?) \\
&last\_playing' = last\_playing \\
&music\_info' = \{music?\} \lhd music\_info
\end{aligned}
$$

图 2.5.6: 管理人员歌曲管理删除事务模式 music_remove

歌曲添加事务需给定歌曲 music？, 并提供歌曲的名称，歌曲的作者，歌曲的主题

music_add

$$
\begin{aligned}
&music\_manage \\
&tilte? : TITLE \\
&author? : \mathbb{F}\, AUTHOR \\
&subjects? : \mathbb{F}\, SUBJECT \\
&\hline
&music? \notin playlist \\
&playlist' = playlist \cup (music?) \\
&unplay' = unplay \cup (music?) \\
&last\_playing' = last\_playing \\
&music\_info' = music\_info \oplus \{music \mapsto m\} \\
&\quad where\ m : SONG \mid SONG.tilte = tilte? \\
&\qquad\qquad\qquad\quad SONG.author = author? \\
&\qquad\qquad\qquad\quad SONG.subjects = subjects?
\end{aligned}
$$

图 2.5.7: 管理人员歌曲管理添加事务模式 music_add

（2）查询事务

查询事务不影响音乐播放的状态

操作模式 Manage_list 可通过用户标志 operator? 查找用户对于账户的操作，查阅结果 list! 是音乐信息的集合

Manage_list

$$
\begin{aligned}
&music\_manage \\
&operator? : PERSON \\
&list! : \mathbb{F}(MUSIC \times SONG) \\
&\hline
&operator? \in manager \cup user \\
&list! = \{c : MUSIC \mid playing(c) = operator? \bullet (c, music\_info(c))\}
\end{aligned}
$$

图 2.5.8: 管理人员歌曲查询事务模式 Manage_list

2.5.2 普通用户事务

普通用户事务包括音乐查询事务，音乐播放事务

所用普通用户事务包含以下特点：可以查询音乐和播放音乐，引入模式 User_trans ,LIB 状态模式不受到改变。

User_trans

$$
\begin{aligned}
&\Delta LIB \\
&id? : PERSON \\
&\hline
&id? \in user \cup manager \\
&\theta LIB' = \theta LIB
\end{aligned}
$$

图 2.5.9: 普通用户人员事务模式 User_trans

查询歌曲作者信息以满足给定描述或者主题信息满足给定的音乐信息集合引入给定描述与歌手信息对应关系和给定描述与主题信息的相关信息

[DESC]

$$
\begin{aligned}
&A\_match : DESC \leftrightarrow AUTHOR \\
&S\_match : DESC \leftrightarrow SUBJECT
\end{aligned}
$$

（1）歌曲查询事务

用户可以通过歌手信息描述查询歌曲，Author_enquriy 根据用户输入的歌手信息描述 d?, 给出关于歌手的所有歌曲信息

Author_enquiry

$$
\begin{aligned}
&User\_trans \\
&d? : DESC \\
&list! : \mathbb{P}\, MUSIC \\
&\hline
&list! = \{m : MUSIC \mid m \in \operatorname{ran} music\_info \wedge (m.author \cap A\_match(d?) \neq \emptyset)\}
\end{aligned}
$$

图 2.5.10: 普通用户查询操作模式 Author_enquiry

用户可以通过主题信息描述查询歌曲，Subject_enquriy 根据用户输入的主题信息描述 d?, 给出关于主题的所有歌曲信息

Subject_enquriy

$$User\_trans$$
$$d? : DESC$$
$$list! : \mathbb{P}\, MUSIC$$

$$list! = \{m : MUSIC \mid m \in \operatorname{ran} music\_info \land (m.subjects \cap S\_match(d?) \neq \emptyset)\}$$

图 2.5.11: 普通用户查询操作模式 Subject_enquriy

用户可以播放已管理的歌曲，Playing_music 进行歌曲播放

Playing_music

$$User\_trans$$
$$out! : \mathbb{P}\, MUSIC$$

$$out! = \{m : MUSIC \mid m \in \operatorname{dom} playing\}$$

图 2.5.12: 普通用户音乐播放模式 Playing_music

2.6 操作模式前置条件

Pre_account_add

$$LIB?;$$
$$id? : PERSON$$
$$playlist? : LIST$$
$$playlist\_title? : TITLE$$

$$id? \in manager$$
$$playlist\_title? \notin \operatorname{ran} list_info$$

图 2.6.1: 前置条件模式 Pre_account_add

Pre_account_remove

$$LIB;$$
$$id? : PERSON$$
$$playlist? : LIST$$
$$playlist\_title? : TITLE$$

$$id? \in manager$$
$$playlist\_title? \in \operatorname{ran} list_info$$

图 2.6.2: 前置条件模式 Pre_account_remove

Pre_music_add

LIB;
$id? : PERSON$
$music : MUSIC$
$title? : TITLE$
$author? : \lhd AUTHOR$
$subjects? : \lhd SUBJECT$

$id? \in manager$
$music? \notin unplay$
$music? \notin playlist$

图 2.6.3: 前置条件模式 Pre_music_add

Pre_music_remove

LIB;
$id? : PERSON$
$music : MUSIC$

$id? \in manager$
$music? \in unplaymusic? \in playlist$

图 2.6.4: 前置条件模式 Pre_music_remove

Pre_Author_enquiry

LIB;
$id? : PERSON$
$d? : DESC$

$id? \in manager \cup user$

图 2.6.5: 前置条件模式 Pre_Author_enquiry

Pre_Subject_enquiry

LIB;
$id? : PERSON$
$d? : DESC$

$id? \in manager \cup user$

图 2.6.6: 前置条件模式 Pre_Subject_enquiry

Pre_Playing

LIB;
$id? : PERSON$

$id? \in \text{ran}\, manager \cup user$
$\text{dom}\, playing \neq \emptyset$

图 2.6.7: 前置条件模式 Pre_Playing

选取前置条件模式 Pre_music_add 进行证明证明

music_add

$music\_manage$
$id? : \mathbb{P}\, PERSON$
$tilte? : TITLE$
$author? : \mathbb{F}\, AUTHOR$
$subjects? : \mathbb{F}\, SUBJECT$

$\exists\, music\_manage;\ \bullet$
$(music? \notin playlist$
$music? \notin unplay$
$playlist' = playlist \cup (music?)$
$unplay' = unplay \cup (music?)$
$last\_playing' = last\_playing$
$music\_info' = music\_info \oplus \{music \mapsto m\}$
$\quad where\ m : SONG \mid SONG.tilte = tilte?$
$\qquad\qquad\qquad\qquad SONG.author = author?$
$\qquad\qquad\qquad\qquad SONG.subjects = subjects?)$

考虑该模式的谓词部分，因为 $music? \notin playlist\ \&\&\ unplay$ 由点规则得到 $playlist' = unplay/playlist \cup (music?)$, 且 $last\_playing = last\_playing$ 故可对 1，2，3 式进行消去

music_add

$music\_manage$
$id? : \mathbb{P}\, PERSON$
$tilte? : TITLE$
$author? : \mathbb{F}\, AUTHOR$
$subjects? : \mathbb{F}\, SUBJECT$

$\exists\, music\_manage;\bullet$
$(music\_info' = music\_info \oplus \{music \mapsto m\}$
$\quad where\ m : SONG \mid SONG.tilte = tilte?$
$\qquad\qquad\qquad\qquad SONG.author = author?$
$\qquad\qquad\qquad\qquad SONG.subjects = subjects?)$

因为 $m.title : TITLE, m.author : AUTHOR, m.subjects : SUBJECT, m : MUSIC$ 并且 $music\oplus \notin music\_info$, 所以 $music\_info' \equiv music\_info \oplus \{music \mapsto m\}\}$, 进行消去。同时，添加的 author,subjects 应当属于尚未添加的属性，故在定义域上添加 $\lhd$, 证明完毕

Pre_music_add

$LIB;$
$id? : PERSON$
$music : MUSIC$
$title? : TITLE$
$author? : \lhd AUTHOR$
$subjects? : \lhd SUBJECT$

$id? \in manager$
$music? \notin playlist$
$music? \notin unplay$

2.7 操作的完整性描述和出错处理

模式 Error

$$
\begin{array}{|l}
\hline
Error \\
\Delta LIB; \\
r! : REPORT \\
\hline
\theta LIB = \theta LIB' \\
\hline
\end{array}
$$

模式 Success

$$
\begin{array}{|l}
\hline
Success \\
rep! : REPORT \\
\hline
rep! :' OK' \\
\hline
\end{array}
$$

模式 Unknown_User

$$
\begin{array}{|l}
\hline
Unknown\_User \\
Error \\
id? : PERSON \\
\hline
id? \notin manager \cup user \\
r! :=' Unknown\_User' \\
\hline
\end{array}
$$

模式 Too_Many_Playlist

$$
\begin{array}{|l}
\hline
Too\_Many\_Playlist \\
Error \\
id? : PERSON \\
\hline
\#(account) \geq Playlist\_Limit \\
r! :=' Too\_Many\_Playlist' \\
\hline
\end{array}
$$

模式 Too_Many_Music

$$
\begin{array}{|l}
\hline
Too\_Many\_Music \\
Error \\
id? : PERSON \\
\hline
\#(playlist) \geq Music\_Limit \\
r! :=' Too\_Many\_Music \\
\hline
\end{array}
$$

添加 M_Error, 描述音乐不在当前歌单内的状态

$$M\_Error \mathrel{\widehat=} [Error \S music? : MUSIC]$$

$$
\begin{array}{|l}
\hline
Music\_Not\_In \\
M\_Error \\
\hline
(music? \notin playlist \land r!) =' Music\_Not\_In') \\
\lor (music? \in playlist \setminus unplay \land \\
r! = 'Music\_Is\_Playing') \\
\hline
\end{array}
$$

图 2.7.1: 模式 Music_Not_In

添加 A_Error, 描述不存在歌单内的状态

$$A\_Error \mathrel{\widehat=} [Error \S playlist? : MUSIC]$$

$$
\begin{array}{|l}
\hline
Playlist\_Not\_In \\
A\_Error \\
\hline
(playlist? \notin account \land r!) =' Playlist\_Not\_In') \\
\hline
\end{array}
$$

图 2.7.2: 模式 Playlist_Not_In

定义音乐播放的完整操作描述为：

$$
\begin{aligned}
T\_Playing \mathrel{\widehat=} & (Music\_Playing \land Success) \\
& \lor Unknown\_User \\
& \lor Too\_Mang\_Music \\
& \lor Music\_Not\_In
\end{aligned}
$$

定义歌单在账户中的完整操作描述为：

$$
\begin{aligned}
Playlist\_In\_Account \mathrel{\widehat=} [& A\_Error \mid playlist? \in account \\
& \land r! =' Playlist\_In\_Account']
\end{aligned}
$$

定义音乐在播放列表中完整描述为：

$$
\begin{aligned}
Music\_In\_Playlist \mathrel{\widehat=} [& M\_Error \mid music? \in playlist \\
& \land r! =' Music\_In\_Playlist']
\end{aligned}
$$

添加歌单操作模式

$$
\begin{aligned}
Playlist\_Add \mathrel{\widehat=} & (playlist\_add \land Success) \\
& \lor Unknown\_manager \\
& \lor Playlist\_In\_Account
\end{aligned}
$$

删除歌单操作模式

$$
\begin{aligned}
Playlist\_Remove \mathrel{\widehat=} & (playlist\_remove \land Success) \\
& \lor Unknown\_manager \\
& \lor Playlist\_Not\_In
\end{aligned}
$$

添加音乐操作模式

$$
\begin{aligned}
Music\_Add \mathrel{\widehat=} & (music\_add \land Success) \\
& \lor Unknown\_manager \\
& \lor Playlist\_In\_account
\end{aligned}
$$

删除音乐操作模式

$$
\begin{aligned}
Music\_Remove \mathrel{\widehat=} & (music\_remove \land Success) \\
& \lor Unknown\_manager \\
& \lor Music\_Not\_In
\end{aligned}
$$

3 规格说明的性质和证明

管理员添加一个有效的歌曲并立即删除，不会导致歌单信息的状态改变

可表示为如下定理：$music\_add \fatsemi music\_remove \| id? \; music? \; title? \; author? \; subjects? \vDash \Xi DB$

记录新模式 $music\_add \fatsemi music\_remove$ 为 AddAndRemove

music_remove

$$
\begin{array}{l}
music\_manage \\
\hline
music? \in playlist \\
playlist' = playlist \setminus (music?) \\
unplay' = unplay \setminus (music?) \\
last\_playing' = last\_playing \\
music\_info' = \{music?\} \lhd music\_info
\end{array}
$$

music_add

$$
\begin{array}{l}
music\_manage \\
tilte? : TITLE \\
author? : \mathbb{F}\, AUTHOR \\
subjects? : \mathbb{F}\, SUBJECT \\
\hline
music? \notin playlist \\
playlist' = playlist \cup (music?) \\
unplay' = unplay \cup (music?) \\
last\_playing' = last\_playing \\
music\_info' = music\_info \oplus \{music \mapsto m\} \\
\quad where\ m : SONG \mid SONG.tilte = tilte? \\
\qquad\qquad SONG.author = author? \\
\qquad\qquad SONG.subjects = subjects?
\end{array}
$$

AddAndRemove

$$
\begin{array}{l}
\Delta DB \\
id? : PERSON \\
music? : MUSIC \\
title? : TITLE \\
author? : AUTHOR \\
subjects? : SUBJECT \\
\hline
\exists\, DB^{+} \bullet (\\
id? \in manager \\
music? \notin playlist \\
playlist^{+} = playlist \cup (music?) \\
last\_playing^{+} = last\_playingmusic\_info\oplus = music\_info \oplus \{music \mapsto m\} \\
\quad where\ m : SONG \mid SONG.tilte = tilte? \\
\qquad\qquad SONG.author = author? \\
\qquad\qquad SONG.subjects = subjects? \\
music? \in playlist^{+} \\
playlist' = playlist^{+} \setminus (music?) \\
unplay' = unplay^{+} \setminus (music?) \\
last\_playing' = last\_playing^{+} \\
music\_info' = \{music?\} \lhd music\_info^{+} \\
)
\end{array}
$$

展开该定理

$$
\begin{array}{|l}
\textit{AddAndRemove} \\
\Delta DB \\
id? : PERSON \\
music? : MUSIC \\
title? : TITLE \\
author? : AUTHOR \\
subjects? : SUBJECT \\
\hline
\exists\, DB^{+}; account : \mathbb{P}\, LIST \\
playlist : \mathbb{P}\, MUSIC \\
unplay : \mathbb{P}\, MUSIC \\
playing : MUSIC \pfun PERSON \\
last\_playing : \mathbb{P}\, MUSIC \pfun PERSON \\
music\_info : \mathbb{P}\, MUSIC \pfun SONG \\
list\_info : \mathbb{P}\, LIST \pfun STORE \ \bullet \ (\\
id? \in manager \\
music? \notin playlist \\
playlist^{+} = playlist \cup (music?) \\
last\_playing^{+} = last\_playing \\
music\_info\oplus = music\_info \oplus \{music \mapsto m\} \\
\quad where\ m : SONG \mid SONG.tilte = tilte? \\
\qquad\qquad\qquad\quad SONG.author = author? \\
\qquad\qquad\qquad\quad SONG.subjects = subjects? \\
music? \in playlist^{+} \\
playlist' = playlist^{+} \setminus (music?) \\
unplay' = unplay^{+} \setminus (music?) \\
last\_playing' = last\_playing^{+} \\
music\_info' = \{music?\} \rhd music\_info^{+} \\
) \\
\end{array}
$$

因为 last_playing'=last_playing ,playing'=playing list_info'=list_info 故可消去
现在证明, 下式不会导致状态的改变

$$
\begin{array}{l}
music\_info' = \{music?\} \rhd (music\_info \oplus \{music \mapsto m \\
\quad where\ m : SONG \mid SONG.tilte = tilte? \\
\qquad\qquad\qquad\quad SONG.author = author? \\
\qquad\qquad\qquad\quad SONG.subjects = subjects?)\} \\
playlist' = playlist \cup (music?) \setminus (music?) \\
unplay' = unplay \cup (music?) \setminus (music?)
\end{array}
$$

因为取并集后再进行删除，可知 $playlist' = playlist, unplay' = unplay$
得证