

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE
ENGINEERING AND TECHNOLOGY

**AN APPROACH TO COPE WITH STATE SPACE EXPLOSION PROBLEM IN
MODEL CHECKING EMBEDDED TV SOFTWARE**

M.Sc. THESIS

Furkan CÖMERT

Department of Computer Engineering

Thesis Advisor: Asst.Prof.Dr.Tolga OVATMAN

DECEMBER 2015

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE
ENGINEERING AND TECHNOLOGY

**AN APPROACH TO COPE WITH STATE SPACE EXPLOSION PROBLEM IN
MODEL CHECKING EMBEDDED TV SOFTWARE**

M.Sc. THESIS

Furkan CÖMERT
504121518

Department of Computer Engineering

Thesis Advisor: Asst.Prof.Dr.Tolga OVATMAN

DECEMBER 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

**AN APPROACH TO COPE WITH STATE SPACE EXPLOSION PROBLEM IN
MODEL CHECKING EMBEDDED TV SOFTWARE**

YÜKSEK LİSANS TEZİ

**Furkan CÖMERT
504121518**

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Yrd.Doç.Dr. Tolga OVATMAN

ARALIK 2015

Furkan CÖMERT, a **M.Sc.** student of **ITU Graduate School of Science Engineering and Technology** student ID **504121518**, successfully defended the thesis entitled “**AN APPROACH TO COPE WITH STATE SPACE EXPLOSION PROBLEM IN MODEL CHECKING EMBEDDED TV SOFTWARE**”, which he prepared after fulfilling the requirements specified in the associated legislations. before the iurv whose signatures are below.

Thesis Advisor : **Asst.Prof.Dr. Tolga OVATMAN**
Istanbul Technical University

Jury Members : **Prof.Dr. Nadia ERDOĞAN**
Istanbul Technical University

Assoc.Prof.Dr. Mehmet Nafiz AYDIN
Kadir Has University

Date of Submission : 26 November 2015
Date of Defense : 21 December 2015

To the love of my life,

FOREWORD

This thesis is written for my master degree in Computer Engineering Graduate Program at Istanbul Technical University. The subject of this thesis is “An approach to cope with state space explosion problem in model checking embedded TV software”.

I started my master education at 2012 and I had started my professional life in Arçelik A.Ş. at 2011. In the beginning, it was hard to work in a company and study at the same time. But I did my best to overcome and then I succeed now.

I would like to thank the following people, without their support this thesis would not have been come up. First, I would like to thank my supervisor Asst. Prof. Tolga Ovatman for giving me valuable advice and support always when I needed. I would also like to thank Onur Dinçer (Director of SW R&D Department in Arçelik), Barış Özbilek (My former Team Leader in Arçelik) and Özgür Çelebican (My current Team Leader in Arçelik) for their encouragement to my master degree.

In 2014, I met Bilge, the love of my life. I dedicate my thesis to her.

Finally, I would like to thank my father Mustafa Cömert, my mother Nurhayat Cömert and My sister Miray Özge Cömert for their constant support during the time I studied.

December 2015

Furkan CÖMERT

TABLE OF CONTENTS

	<u>Page</u>
FOREWORD	ix
TABLE OF CONTENTS.....	xi
ABBREVIATIONS	xiii
LIST OF TABLES	xv
LIST OF FIGURES	xvii
SUMMARY	xix
ÖZET.....	xxi
1. INTRODUCTION.....	1
2. LITERATURE REVIEW.....	3
3. SOFTWARE VERIFICATION.....	7
4. MODEL CHECKING	9
4.1 Model Checking Concept.....	9
4.2 Model Checking Basic Procedure	10
5. NUSMV	13
5.1 Kripke Structure	13
5.2 NuSMV Syntax	13
5.2.1 Declaring state variables	13
5.2.2 Assignments	14
5.2.2.1 Initialization assignment	14
5.2.2.2 Progression assignment	14
5.2.2.3 Immediate assignment.....	14
5.2.3 Expressions	15
5.2.4 Arrays.....	15
5.2.5 Syntax for ASSIGN statements.....	16
5.2.6 ASSIGN statement restrictions	16
5.2.6.1 Single assignment rule	16
5.2.6.2 Circular dependencies rule.....	17
5.2.7 Modules.....	17
5.3 CTL Model Checking with NuSMV	18
5.3.1 Computational tree logic in NuSMV	18
5.3.2 Linear temporal logic in NuSMV	20
6. MODELLING TV SOFTWARE USING SMV	23
6.1 Audio Source Control Module of TV	23
6.2 NuSMV Module of Audio Source Control Module.....	25
6.3 CTL Specifications To be Checked	38
7. TV LOGS	39
8. TEST LOG BASED SPECIFICATION OF USER BEHAVIOR	41
8.1 Direct Log Based Specification	41
8.2 Injecting Nondeterminism to Log Based Specification	43
8.3 Combining Log-Based Specification	45

9. RESULTS.....	51
10. CONCLUSION.....	55
REFERENCES	57
APPENDICES	61
APPENDIX A	61
APPENDIX B.....	62
CURRICULUM VITAE	93

ABBREVIATIONS

ARC	: Audio Return Channel
AVL	: Auto Volume Levelling
AT	: Atomic Proposition
ATV	: Analog Television
BBT	: Black Box Testign
BDD	: Binary Decision Diagram
CAS	: Conditional Access System
CMU	: Carnegie Mellon University
CTL	: Computational Tree Logic
DVB-C	: Digital Video Broadcasting – Cable
DVB-S	: Digital Video Broadcasting – Satellite
DVB-T	: Digital Video Broadcasting – Terrestrial
DTV	: Digital Television
FSM	: Finite State Machine
HDMI	: High Definition Multimedia Interface (TV input source)
HP	: Headphone
HW	: Hardware
LTL	: Linear Temporal Logic
PC	: Personal Computer (TV input source)
RC	: Remote Controller
RCE	: Remote Controller Emulator
SMT	: Satisfiability Modulo Theories
SMV	: Symbolic Model Verifier
SSE	: State Space Explosion
STB	: Set-Top-Box
STS	: State Transition System
SW	: Software
TS	: Transition System
TV	: Television
UML	: Unified Modelling Language
USB	: Universal Serial Bus (TV media input source)

LIST OF TABLES

	<u>Page</u>
Table 5.1 : The syntax of CTL formulas recognized by NuSMV [35]	19
Table 5.2 : The syntax of LTL formulas recognized by NuSMV [37]	21
Table 6.1 : Specifications	38
Table 7.1 : Log keys and corresponding RC buttons	39
Table 8.1 : Example log content.....	42
Table 8.2 : RC keys and corresponding RC buttons	42
Table 8.3 : RC keys and corresponding keystroke frequencies	45
Table 8.4 : RC keys and corresponding RC buttons	45
Table 8.5 : Example log content.....	47
Table 8.6 : RC keys and corresponding frequencies	47
Table 8.7 : RC keys and corresponding RC buttons	48
Table 8.8 : The RC buttons which corresponds to the transitions in Figure 8.6	49
Table 9.1 : Direct log based approach results	52
Table 9.2 : Injecting Non-determinism to log based approach results.....	52
Table 9.3 : Combining log based approach results	53

LIST OF FIGURES

	<u>Page</u>
Figure 4.1 : Schematic view of the model checking approach [31].....	9
Figure 5.1 : The block presentation	18
Figure 6.1 : State diagram of the source menu navigation.....	34
Figure 6.2 : State diagram of the dvb-t channel navigation.....	35
Figure 6.3 : State diagram of the dvb-c channel navigation	36
Figure 6.4 : State diagram of the dvb-s channel navigation.....	37
Figure 8.1 : RC Key (transition) is assigned just between the previous and the next states.....	42
Figure 8.2 : The corresponding RC Key transition system of the output user agent module.....	43
Figure 8.3 : RC Key (transition) is assigned between the previous and the next states and this RC key (transition) is also be assigned as a loop transition in the same state to provide a non-deterministic transition	44
Figure 8.4 : Generated transition system for “Injecting Non-determinism to Log Based Specifications” approach for the log file on the Table 8.1.....	46
Figure 8.5 : Each RC Keys in the logs are assigned as a transition between the corresponding states of the other log file.	48
Figure 8.6 : Generated transition system for “Combining Log Based Specifications” approach for the log files on the Table 8.1 and Table 8.5.....	48

AN APPROACH TO COPE WITH THE STATE SPACE EXPLOSION PROBLEM IN MODEL CHECKING EMBEDDED TV SOFTWARE

SUMMARY

Digital technology is developing rapidly. Thus the demand of new features for the TVs is increasing speedily. Besides, this increasing demand causes stronger competition. As a result of strong competition and increasing demand for the new features, embedded TV software of current TV sets is getting more complicated. Therefore, the test and verification process of the embedded TV software is getting harder.

Logic of model checking is to algorithmically check if a model satisfies a specification. A model generally expressed as a directed graph including vertices and edges. The vertices specify the states of the model and the edges specify the state transitions due to a given condition.

It is a widely accepted practice to use model checking in verification of embedded systems. On the other hand, it is seriously affected by the exponential increase in the number of states during the verification process. Modelling a fully non-deterministic TV-user is the one of the main reason for the exponential increase in the number of states. For the purpose of shrinking the state space being observed during the model checking process of TV, three methods are proposed in this thesis. TV logs are used to generate partly non-deterministic TV-user (user agent) models for all three methods that are proposed. The TV logs includes the Remote Controller (RC) keystroke histories recorded during the manual testing of the TV.

The first method is “Direct Log Based Specification”. In this method, the user agent is designed directly from the TV log. The number of states are decreased in this method whereas the user agent model became fully deterministic.

The second method is “Injecting non-determinism to Log Based Specifications”. In this method the first method is improved and a more non-deterministic user agent is produced according to the given TV log.

The Third method is “Combining Log Based Specifications”. In this method we used two different TV logs to construct a more non-deterministic user-agent.

Our results show that by using partly non-deterministic user agents we can significantly decrease the verification time of certain safety and liveness properties.

GÖMÜLÜ TV YAZILIMI MODELİNİN KONTROLÜNDE ORTAYA ÇIKAN DURUM PATLAMASI PROBLEMİNE ÇÖZÜM YAKLAŞIMI

ÖZET

Üniversitelerde yazılan makale sayıları gün geçtikçe daha da artıyor. Bunun yanında bilime ayrılan önem ve bütçe de gittikçe artıyor. Dolayısıyla artık neredeyse hergün bilimin ürünü olan yeni bir bilgi üretiliyor. Bunun sayesinde teknoloji de her zaman olduğundan daha da hızlı bir şekilde gelişmeye devam ediyor. Teknolojinin gelişmesi tüketici elektroniği sektörünün arkasındaki rüzgarı da arttırarak kullandığımız elektronik ürünlerin daha fonksiyonel ve işe yarar olmasına katkıda bulunuyor.

2007 yıllarında daha tüplü TV'ler kullanılıyorken, kimyasal ve elektronik araştırmalar neticesinde önce LCD ve Plazma daha sonra da LED TV'ler piyasaya sunuldu. Bu teknolojiler sayesinde TV panel kalınlıklarında dramatik bir düşüş meydana geldi ve TV'lerin gereksiz yer israfı bitti.

Görüntüleme (display) teknolojilerindeki araştırmalar sonucu 3 boyutlu filmler hem sinemalarda hem de evde TV'lerde izlenmeye başladı. Bu konudaki araştırmalar ile TV'lerin 3 boyutlu yayınlardaki görünüş derinliği ve görüntü kalitesi yazılan yeni algoritmalar ve görüntüleme tekniğindeki gelişmeler ile daha da arttı.

İnternet kullanımının artması ve yazılımsal gelişmeler sonucu internet TV'lerde de yer almaya başladı. Bu sayede TV'lerden internet sitelerine girilebilmeye, videolar filmler izlenebilmeye başlandı.

Bluetooth teknolojisindeki gelişmeler ile TV'ler A2DP protokolünü kullanarak diğer bluetooth hoporlor cihazlara kablosuz ses iletmeye, AVRCP protokolünü kullanarak bu cihazlardan geribildirim almaya ve HID protokolünü kullanarak oyun konsolu vb... cihazlar ile kumanda edilmeye başlandı.

TV'lere birden fazla tuner bağlanarak ve özel gözlükler kullanılarak iki farklı yayının aynı anda izlenebilmesi sağlandı.

Full HD TV'lerin 4 katı çözünürlüğe sahip 4K paneller kullanılmaya başlandı. 2015 CES fuarında Sharp firması 8K panele sahip TV'sini tanıttı.

Yeni teknolojiler, firmalar arası rekabet ve artan talep doğrultusunda TV fonksiyonallitesi çok gelişti ve dolayısıyla TV'lerin gömülü sistem yazılımı çok büyüdü ve karmaşılaştı. Dolayısıyla TV gömülü yazılımı test ve doğrulama sürecide daha da zorlaştı.

Bu tez içerisinde TV gömülü yazılımı test ve doğrulama sürecinin hızlandırılabilmesi amacıyla TV gömülü yazılım bileşenlerinin modellenerek, model üzerinde yapılacak sorgular ile test ve doğrulama sürecinde bir alternatif oluşturulması düşünüldü. Dolayısıyla bu tez içerisinde TV gömülü sistem yazılımı ses ile ilgili kısımların bir kısmı SMV dili kullanılarak modellendi.

Model içerisinde TV'lerin genellikle kullanıcı ses menüsü altında bulunan bileşenler bu tez çalışması için kullanıldı. Model içerisinde gerçek bir kullanıcının modellenebilmesi amacıyla deterministic olmayan bir kullanıcı (a nondeterministic user agent) modellendi. Deterministik olmayan kullanıcı modelimiz TV kumanda şablonunda belirtilen 35 adet tuşun herhangi birine herhangi bir zamanda basabilecek şekilde modellendi. Bununla birlikte modelde kullanılan TV ses bileşenleri şunlardır:

- Kullanıcı seçimi doğrultusunda TV sesinin iyileştirilmesi için kullanılan “Sound Mode” bileşeni modele eklendi.
- Gece geç vakitlerde kullanıcıların komşularını rahatsız etmeden TV izlemelerini ses seviyesinin otomatik minimize edilmesi ile sağlayan “Night Mode” bileşeni modele eklendi.
- Bluetooth A2DP protokolü kullanılarak harici cihazlara kablosuz ses gönderme işlevini sağlayan “Bluetooth” bileşeni modele eklendi.
- TV Ses performansının, TV'nin ayakta konumunda bulunmasına göre ya da duvara asılı konumda bulunmasına göre maksimize edilmesini sağlayan “TV Location” bileşeni eklendi.
- HDMI kablosu üzerinden ters yönde ses gönderimini sağlayan “ARC” (Audio Return Channel) bileşeni modele eklendi.
- TV Scart ses çıkışı seviyesinin ayarlanmasına olanak sağlayan “Scart Volume” bileşeni modele eklendi.
- Birden fazla kanala sahip ses bilgisinin kaliteli bir biçimde harici ses sistemlerine aktarılmasında görev alan “Optic Out” bileşeni modele eklendi.
- TV Ses ayarlarının fabrika ayarlarına ayarlanabilmesine olanak sağlayan “RestoreToDefault” bileşeni modele eklendi.
- Modele eklenen TV giriş kaynakları şunlardır: HDMI (High Definition Multimedia Interface), SCART (Radio and Television Receiver Manufacturers' Association), Component (YPbPr - Component Video), Cable (DVB-C), Terrestrial (DVB-T), Satellite (DVB-S), ATV(Analog TV), USB girişi (Universal Serial Bus) ve PC girişi.
- Kanal geçiş mekanizması, sayısal ses yükselticinin, ses yollarının ve ses çözücülerinin ayarlanması ile ilgili kısımlar da modele eklendi.

Tasarlanan model 48 GB hafızaya sahip 12 çekirdekli güçlü bir server bilgisayar üzerinde denendi. Modelden yaratılan durum uzayını belirten BDD'nin (Binary Decision Diagram) çapı bilgisini de verebilme yeteneğine sahip olan NuSMV Model denetçisi kullanıldı. Bu tez de çap ve model denetimi süreleri bilgileri üzerinde çalışma yapılarak BDD çapının ve model denetim sürelerinin azaltılması sağlandı.

Bu tez de, modelin 2 adet Canlılık (Liveness) ve 2 adet Güvenlik (Safety) özellikleri kontrol edildi.

Fakat NuSMV üzerinde koşulan sorguların çok fazla zaman alabildiği görüldü. Bu saatler bazından eğer model biraz daha büyütülürse günler bazına kadar bile

çıkabilmektedir. Bunun nedeni durum uzayı patlaması (State-Space Explosion) problemidir.

“State-Space Explosion” problemini çözmek amacıyla 3 adet yöntem geliştirildi ve bu tez de tanıtıldı. Bu problemin çözümünü içeren 3 adet yöntem içerisinde beşeri testçilerin manual testlerindeki TV kumanda tuşları geçmişini içeren TV logları üzerinde çalışıldı.

Faklı model TVlerin kumanda tuş içerikleri arasındaki farklılıklardan dolayı bir şablon üzerinde karar kılındı ve bu şablon kullanıldı. Şablon içerisinde 35 adet TV kumanda tuşu bulunmaktadır. TV kumanda tuşu geçmişlerini içeren loglar şablonda belirlenen 35 farklı tuştan herhangi birilerini içerebilmektedir.

Bu üç adet yöntem içerisinde, model içerisinde modellenmiş olan ve deterministik olmayan kullanıcı temsilinin yerine TV logları kullanılarak deterministik olmayan bir kullanıcı temsili oluşturulmaya çalışılmış ve sorgu süresi ve BDD çapı düşürülmeye çalışılmıştır.

Tez içerisinde “State-Space Explosion” problemine çözüm olarak sunulan ilk yöntem “Direct log based specification” yaklaşımıdır. Bu yöntem içerisinde temsili kullanıcı direkt olarak TV logları kullanılarak oluşturulmaktadır.

Bu yöntem içerisinde, log dosyası içerisinde belirtilen her kumanda tuşu bilgisi, kullanıcı modülünde ki herbir farklı durum geçişine dönüştürülmektedir.

Bu yaklaşımda log dosyaları kullanıcı modülüne direkt olarak dönüştürülmektedir. Dolayısıyla bu yaklaşımın avantajı olarak sorgu süresi ve durum uzayı BDD çapı düşürülmüştür. Fakat bu yaklaşımın dezavantajı olarak, modelde kullanılan kullanıcı modülü deterministik olmama özelliğini tamamen kaybederek tamamen deterministik olmuştur. Dolayısıyla kullanılan kullanıcı temsili, beşeri kullanıcı özelliklerinden tamamen uzaklaşmıştır.

“Direct log based specification” yaklaşımının dezavantajı olan tamamen deterministik kullanıcı temsilinin, deterministik olmayan bir kullanıcı temsiline dönüştürülmesi ve dolayısıyla beşeri bir kullanıcı davranışına yaklaşması amacıyla bu tezde ikinci yöntem “Injecting Non-determinism to Log Based Specification” geliştirildi ve tanıtıldı.

Bu yaklaşımda log dosyası içerisindeki herbir tuş bilgisinin log dosyası içerisinde tekrar edilme frekansı hesaplanıyor. Her bir farklı tuş bilgisi eğer frekansı 1 ise, her bir farklı durum geçişine dönüştürülüyor. Eğer tuş bilgisinin frekansı 1’den yüksek ise, o tuş bilgisi deterministik olmayan bir durum geçişine dönüştürülüyor. Bu mantık ile deterministik olmayan bir kullanıcı temsili oluşturulmaya çalışıldı.

Log dosyası kullanılarak deterministik olmayan bir kullanıcı temsili modellenmesi ve ayrıca sorgu süreleri ve BDD çapının düşürülmesi “Injecting Non-determinism to Log Based Specification” yaklaşımının avantajı olarak belirtildi.

Farklı TV kullanıcıların kumanda tuşuna basma geçmişlerini içeren log dosyalarından oluşturulan kullanıcı temsili modüllerinin birleştirilmesi ile elde edilen

deterministik olmayan kullanıcı temsilinin modellenmesi içeren son metod “Combinin log based specifications” bu çalışmada geliştirildi ve bu tez de tanıtıldı.

Bu yaklaşımda, iki adet log dosyasının içerisindeki tuş bilgilerinin frekansları kendi log dosyaları bazında hesaplandı. Herbir tuş, frekanslarına göre, diğer log dosyasından oluşturulan modelin durumlarına erişebilecek şekilde, deterministik olacak şekilde ve deterministik olmayacak şekilde durum geçişlerine modellendi.

Tez de tanıtılan bu yaklaşım ile birlikte kullanıcı temsili modülünün deterministik olmama seviyesi daha da arttırıldı. Ayrıca, model üzerinde ki sorgu tamamlanma süresi ve BDD çapı düşürüldü.

Bu tez içerisinde State- Space Explosion problemine çözüm olarak 3 adet yöntem geliştirilmiş ve sonuçları karşılaştırılmıştır.

1. INTRODUCTION

Televisions and set-top-boxes are the most common examples to consumer electronic devices which are greatly affected by the advances in the embedded software technology. The technology behind TV systems is improved rapidly for twenty years. The functionality and the user interfaces of the today's TV systems are much better and useful according to the TV systems which was manufactured two decades ago. With the introduction of internet connectivity to TV sets and development of smart TV's the development challenges behind TV software is continuously converging with the software development for web and the cloud [1] [2] [3].

As a result of increasing demand for the new features and the rapidly growing challenges in TV software development, embedded TV software of current TV sets is getting more complicated. Therefore, the test and verification process of the embedded TV software is getting harder and more complicated. Due to the strong competition in the market and the demand for the cutting edge technology TVs also increase that produce a push in developing software with fewer bugs in shorter times.

Model checking [4][5] is a formal methodology for verification of software, performed by exploring all the possible system states in brute force manner. It examines all possible system scenarios that a system may go through in order to check if a certain property holds in all or some of the scenarios.

Model checking also lets the verification be performed at the early stages of the software development. Meanwhile the correctness of the verification is dependent to the quality of the implementation of the model. Because any verification using model-based techniques is only as good as the model of the system. Even though the main strength of model checking is the ability to perform a full coverage of the system under test to seek for properties to be checked, it can also be seen as the main weakness of the approach due to the well-known state space explosion problem [6].

A method is proposed to fight with the state space explosion problem that can be experienced during the model checking of a small portion of a real-world TV system

software in this thesis. In this method, the former testing logs are processed to create user agent modules automatically. The User agent modules are used to inject into the models to be checked. Instead of using fully non-deterministic user agents that may cause state space explosion problem in the model, the various, especially critical, user testing scenarios are extracted and processed to create new user agent modules.

Initially a small section of the audio source control related real-world embedded TV software is modeled for this study. NuSMV model checker [7], which is one of the widely used model checker, is used in this study. Afterwards some basic properties are verified in CTL while measuring the process time that is required to complete the model checking and Binary Decision Diagram (BDD), which is used by NuSMV to represent the state space during the model checking process. The proposed method is applied to create new scenarios and inject them to the model checking process as user agent modules. Consequently, the state space is shrunked in a considerable amount by applying the approach. Additionally, the time that is required to complete the model checking is reduced significantly.

The approach which is presented in this thesis does not require any intervention to the state space production or exploration phases. Thus it can easily be adapted to different model checkers regardless of the type of state space representation strategy being used.

2. LITERATURE REVIEW

State space explosion is one of the main issue in model checking [8] [9] and it is widely studied by model checking community because of that.

AnttiValmari [6] writes one of the earliest studies that contain a detailed explanation of the many methods used to reduce the state space. In his work, Valmari examined the topic in two categories, namely a more general reduction strategies section and advanced state space methods section examining approaches that hold practical significance. The five reduction strategies examined in the paper: transparent construction-time reduction, guided construction-time reduction, preprocessing the model, packed state space and on-the-fly verification.

Recent studies on coping with state space explosion techniques include extending and improving former approaches like sweep-line state space exploration to obtain more memory efficient methods like in Kurt Jensen et al.'s study [10]. Another active area that is leveraged in coping with the problem is state-based model slicing [11] approach where program slicing approaches [12] are applied to models to be checked in order to be able to parallelize model checking process. Similar parallelization approaches exist that works either by leveraging multi-processor systems [13] or by using distributed/cloud computing based strategies [14].

The proposed approach in this thesis, is a model preprocessing method where the model components that represent the user behavior are simplified by reducing the amount of non-determinism. The proposed approach reduce the state space by controlling the amount of non-determinism present in the user model, just like applying abstractions on the models to be verified. This control is automatically inferred by processing the already present test scenarios being applied in testing TV software.

Most of the automatic TV testing studies in the literature prefer black-box or grey-box testing approaches rather than model checking [15][16][17][18][19]. These methodologies focus in performing the tests automatic rather than using people in the

test phase or increase the efficiency of automatic tests being applied. In this thesis, it is also attempted to increase the efficiency of verification studies by leveraging already present test scenarios. Pekovic et al. presented a methodology for automatic functional verification of STB Conditional Access System (CAS) [15]. A pop-up detection instead of manual test phases is used in their system. Markande et al. proposed a methodology for automating the validation of the functionalities in TV to overcome manual validation difficulties [16]. Their work helps to verify audio signals during test automation process and to reduce the time required in validation. Kastelan et al. present a system for efficient testing and functional verification of the CAS in digital TV sets made by different manufacturers [17]. Their work extends the earlier work in Pekovic et al.'s study [15]. Kovacevic et al. proposes a framework, which is much more extensive, and complete framework for automated testing of Set-top boxes (STB) using Black-box testing (BBT) methodology [18]. Their proposed system uses digital image analysis similar to Pekovic et al.'s study [15], for functional failures detection in STB devices. An approach like Model checking can be used as depicted in Marinkovic et al.'s study [19] to verify some properties about the states of a digital TVs. They proposed a method for creating operational profiles that is used for generation of test cases suitable for execution in BBT system.

Murphy et al. presented an automated approach for generating and then reducing a set of test cases [20]. Their approach consists of two steps. The first step is to identify a set of test cases that covers all function sub-paths in the program under test. The second step is to remove the test cases that only cover sub-paths already covered by other tests. It might not be always possible to obtain a test suite that covers the entire state space that can be reachable. Additionally reducing the test cases with applying the second step may damage to the full coverage. In this thesis, a user agent that is able to click a button non-deterministically to obtain a full coverage is used in the model. Kukolj et al. presented a methodology of test case selection and prioritization in testing embedded system to get the same coverage but with fewer test cases [21]. As opposed to this work, we use a nondeterministic user agent for full coverage.

Tekcan et al. proposed an approach for automatic user driven test-case generation for Digital TV (DTV)/STB functional verification [22]. The DTV consumer user profiles are modeled then estimated through suitably designed subjective polls. In the poll

participants are asked to test particular DTV functions. In this thesis, profiles are constructed from the DTV logs automatically (that include the Remote Control behaviors of the users) instead of running user polls as in the approach [22].

The usage of Remote Controller (RC) navigation in the TV tests is also present in the literature. Marinkovic et al. propose an approach for protocol independent Remote Controller Emulator (RCE) to apply transition from one state to another state in the DTV/STB tests [23].

Even though the traces are used from conventional TV software testing approaches, the main contribution is on model checking of TV software in this thesis. In the literature, model checking TV software is not an uncommon practice, however it is generally regarded as a case study in embedded software verification. A typical example is the study of Cordeiro et al. where a model checker is implemented by modifying and extending SMT-based bounded model checking approaches [24].

Another study that applies model checking on embedded TV software is by Dimitra Giannakapoulou and Jeff Merge [25]. This study is not directly related with the state space explosion problem but it presents an approach to model check “fluent-based” linear temporal logic properties on a case study which consists of a fragment of TV software.

A study that shares a number of similar points with in this thesis is performed by Stefania Gnesi and Diego Latella where UML models are transformed to transition systems and then model checked [26]. UML models that are being transformed in this study also do have fully non-deterministic user models where later in the study they have been replaced by simpler user models. This study focuses on model-to-model transformation whereas this thesis mainly focuses on extracting simpler user models automatically from legacy test scenarios.

Wang Ji et al. present a method for slicing hierarchical automata with respect to properties to be verified [27]. On the contrary, the model is not sliced in this thesis; User agent is created in the model according to the TV log files to attack the state space explosion problem.

Leiter and Heaven present a formal framework for automated specification synthesis [28]. They design the specifications as Deontic IO Automata whereas in this thesis, The specifications are defined according to the behavior type of the user agent in the

model and verified under specific scenarios (automatically created by using the TV logs) in this thesis.

Considering all the studies that were performed on testing and verification of embedded TV software, this thesis contributes to the model checking area by proposing a state space reduction method that can be used without modifying the model checker and state space representation technique. The proposed method in this thesis uses legacy test scenarios to automatically produce new scenarios and user model agents which are not fully non-deterministic but does include a controllable amount of non-determinism that can be used to shrink the state space for acceptable verification times.

3. SOFTWARE VERIFICATION

The most generally utilized methods for the purpose of verification of a software are testing and peer reviewing. The software code is examined by the programmer who has not been included in the development cycle of the software project which is under survey for the purpose of peer reviewing process. The code is not executed in this process. Peer reviewing is a powerful method. A vast majority of the bugs can be caught by the peer reviewing process in empirical studies. Then again, the empirical studies demonstrate that the subtle errors and the algorithm defects are difficult to catch by the peer reviewing process because of its non-active nature.

A large amount of the total cost of the software project is left to spend for testing. Thus the testing is one of the most important part of a software project. Testing procedure is not the same as the procedure of peer reviewing. In the peer reviewing process the code is examined manually by the programmers. However, the code is compiled and run for the purpose of testing process. The actual advantage of testing is that it could be exerted to any kind of Software.

In practice, it is not achievable to test all execution paths of the software. Just a little piece of these paths can be tested because of that reason. As a result, this process will not be completed. Because of that, the only output of the testing procedure is indicating the existence of the defect(bug) not the nonexistence of it. This is the most important disadvantage of the testing procedure.

Additionally it is hard to make a decision for the satisfactoriness about the density of the errors that has been found. Practically, the found errors may not cover all the defects of the software yet. Test may need to continue to cover a majority of the errors. It is hard to decide to stop testing.

In the development phase of a software project, There may be so many different type of defects. It is important to catch different type of the defects in order to make output more reliable. Thus it is recommended to use testing and peer review together in the development period of any software project.

4. MODEL CHECKING

4.1 Model Checking Concept

Functionality and features of the TV sets are growing rapidly. This growth on functionality and features in TV sets cause more complex embedded TV software. The more functionality and features cause the more time for the verification process. The cost and the time spent for the verification process are growing greatly. As a result, the strategies are created to decrease the spent time and facilitate the process of verification. Additionally, the coverage of the verification process is also expanded by these strategies. Formal methods provide an early execution of verification while the software project is in the development phase. Thus the spent time for the verification process decreases and the project is verified in a more effective way.

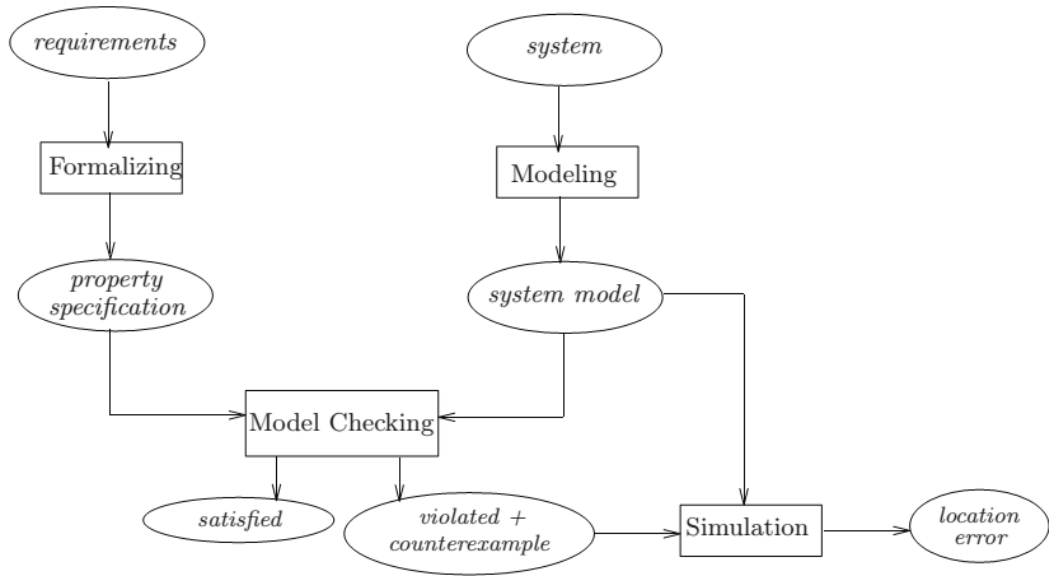


Figure 4.1 : Schematic view of the model checking approach [31].

Initially, the finite state model of the system is constructed in order to check the model of the system. The design of the model is vital for the correctness of the model checking. If the model is designed poorly, the model checking process can not be successful.

In the Model Checking concept, the given specification is checked for the model. Model checker tools are investigated for this purpose. NuSMV model checker tool is used in this thesis. The model checker examines all the paths for the given specification. If the model meets the specification, the model checker results with a positive output. If the specification is not valid for the model (if a state which violates the specification is caught), the model checker results with a counterexample path.

4.2 Model Checking Basic Procedure

Model checking process is composed of the followings:

- Design phase of the model

Model of the system is designed and constructed for the purpose of model checking proses. The quality of the model checking is mainly depends on this phase. Because the model specifies the behavior of the system. If the model is designed in an incorrect way, the results of the model checking process is not correct.

- Execution phase

Model checker is executed to check all the related paths of the system for the given specification.

- Result phase

The possible outputs of the model checker are specified below:

- The model meets the specification:
 - Model checker give positive output.
- The specification is not valid for the model(a state which violates the specification is caught)
 - There may be some types of errors:
 - Modeling error:
 - The model of the system is constructed in a poor way. The model does not represent the behavior of the system. Thus the model should

be corrected in an accurate way in order to re-check the model.

- Property error:
 - The specification does not represent the actual desired requirement to be demanded to check.
- The model is too large to be checked
 - There exist state-space explosion problem. To avoid this problem state shrink algorithms should be developed and applied to the model.

5. NUSMV

Model checking is an approach based on a finite state model of a system and properties requested to be checked for this model. To run model checking automatically, some model checking tools are designed and created. The first constructed BDD [33] based symbolic model checker is SMV. SMV checks every possible path of the model of the system to verify the requested specification for this model. SMV model checker is improved and extended and NuSMV [32] is emerged. NuSMV is developed in Carnegie Mellon University (CMU) and Istituto per la Ricerca Scientifica e Tecnologica (IRST). NuSMV is a BDD based software program that is created to check the finite state model of the system according to a given specification.

5.1 Kripke Structure

Kripke structure is defined by Kripke [34]. The model of the system requested to be checked is a kripke structure. The main aim of the NuSMV model checker is to verify the model (kripke structure) according to the given specification.

Definition 5.1.1 [34]: A Kripke structure $K = (I, S, R, L)$ for a finite set of variables V_Σ is given by a finite set of states S , a set of initial states $I \subseteq S$, a transition relation $R \subseteq S \times S$, and label function $L: S \rightarrow 2^{V_\Sigma}$ that maps each state to a set of variables.

5.2 NuSMV Syntax

5.2.1 Declaring state variables

The content of the Kripke Structure is defined by the state variables. Enumerative, bounded integer and bool are included. Boolean is constructed with TRUE and FALSE symbols.

- A boolean example:

- a : boolean;
- An enumeration example:
 - days : {monday, tuesday, wednesday, thursday, friday, saturday, sunday};
- A bounded integer example (intervals):
 - i : 1..255;

5.2.2 Assignments

5.2.2.1 Initialization assignment

To set the initial value of a variable, Initialization assignment is used.

- An example:

ASSIGN

Init(days) := monday;

Note: To start a variable nondeterministically, no initialization assignment is used for it.

5.2.2.2 Progression assignment

Models includes state transitions. To provide transition of states, the values of the variables should be changed. Progression assignment is used for this purpose.

- An example :

ASSIGN

next(days) := tuesday;

Note: To evolve the states nondeterministically, no “next()” assignment is used for it.

5.2.2.3 Immediate assignment

The current values of the variables are constrained by using Immediate assignment

- Examples

ASSIGN

days := wednesday ;

or:

DEFINE

days := wednesday ;

5.2.3 Expressions

Expressions are used to change the values of the states according to any kind of logic. The logic is defined by using these expressions below.

- Arithmetic operators: +, -, *, /, mod, -(unary),
- Comparison operators: =, !=, >, <, <=, >=
- Logic operators: &, |, xor, !, - >, < - >
- Set operators: {v₁, v₂, ... v_n }, in, union
- Conditional Expression:

```
Variable := case
                a1: b1;
                a2: b2;
                ...
                TRUE: bn;
            esac
```

There exist a “guard” in the conditional expression example. The guard is ”TRUE”. The “TRUE” guard in this example behaves like an “else” statement in C language. If there exist more than one “TRUE” guards, the initial “TRUE” guard specify the end value.

5.2.4 Arrays

Arrays can be used in NuSMV language. The types and the value constraints are defined for the arrays at the definition (VAR) part of the code.

An example:

c : array 1..11 of boolean;

Note: Array includes 11 elements (from 1 to 11) and the type of each element is boolean.

Another example:

d : array 0..6 of array 3..6 of boolean;

Note: Array includes 9 elements (from 0 to 6). Each element of this array includes an array of 3 elements (from 0 to 3). Type of each element of this array is Boolean.

5.2.5 Syntax for ASSIGN statements

An example syntax for assigning the values to variables in NuSMV is presented below:

```
init (<variable>) := <initial_value>
```

```
next(<variable>) := <next_value>
```

Example usage:

VAR

```
b:boolean;
```

```
c:boolean;
```

ASSIGN

```
init(b):=FALSE;
```

```
init(c):= FALSE;
```

```
next(c):= case
```

```
    b=FALSE:TRUE;
```

```
    TRUE:FALSE;
```

```
esac
```

No next assignment specifies nondeterministic value change.

5.2.6 ASSIGN statement restrictions

To avoid conflicts at assignments, some rules are designed. If these rules are obeyed, assignment conflicts are prevented.

5.2.6.1 Single assignment rule

In a program, the assignment for a variable is applied for once. It called “Single Assignment Rule”. Possible conflicts at the assignment are prevented by this rule.

- A Legal example:

```
init(a):=FALSE;
```

```
next(a):=TRUE;
```

- Another Legal example:

```
next(a):=TRUE;
```

```
init(a):=TRUE;
```

```
next(a):=FALSE;
```

- An Illegal example:

```
next(a):=TRUE;
```

```
next(a):=FALSE;
```

```
next(a):=TRUE;
```

```
next(a):=FALSE;
```

5.2.6.2 Circular dependencies rule

There must not be cycles in a variable's dependency graph which are separated by delays.

5.2.7 Modules

A model that is written in NuSMV language may include one or more modules. These modules may be initialized in parent modules. The “VAR” declaration includes the initialization of child modules in a parent module. The NuSMV specifications are specified In the main (top) module.

Initialization via dot notation provide visibility of its (Models) variables in the parent module.

- Example:

```
MODULE mod
```

```
VAR
```

```
    output : 0..3;
```

```
ASSIGN
```

```
    next(output) := output*3;
```

```

MODULE
VAR
    k1 : mod;
    k2 : mod;
    result : 0..81;
ASSIGN
    result:= k1.output * k2.output;

```

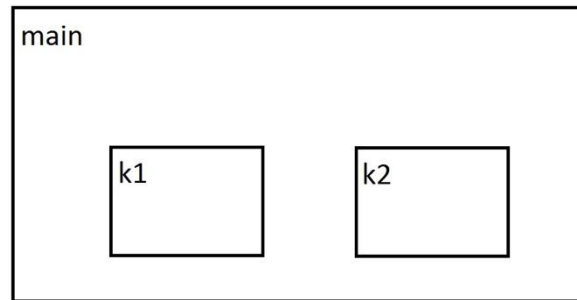


Figure 5.1 : The block presentation

5.3 CTL Model Checking with NuSMV

The basic purpose of a model checker is to check the model due to the requested properties. These properties are designed to verify the model of the system. Two different temporal logics can be used for designing the properties. Computational Tree Logic (CTL) and Linear Temporal Logic (LTL). NuSMV checks the model due to the properties and conclude with a response. If a property is valid for the model, NuSMV returns a positive response, otherwise constructs a counterexample.

5.3.1 Computational tree logic in NuSMV

The model of the system, which is requested to check, is designed like an infinite tree structure in this logic. The FSM states are designed as nodes of the tree. The future state transitions are not certain. The designed specification in this logic starts with “SPEC” and follows “ctl_expression” specified as below:

SPEC <ctl_expression>

The semantics of CTL operators is described in below [36]:

- **EX** p is true in a state s if there exists a state s' such that a transition goes from s to s' and p is true in s' [36].

Table 5.1 : The syntax of CTL formulas recognized by NuSMV [35]

Simple Expression	A Simple Boolean Expression
!ctl_expr	logical not
ctl_expr&ctl_expr	logical and
ctl_expr ctl_expr	logical or
ctl_exprxorctl_expr	logical exclusive or
ctl_exprxnorctl_expr	logical NOT exclusive or
ctl_expr ->ctl_expr	logical implies
ctl_expr<->ctl_expr	logical equivalence
EG ctl_expr	exist globally
EX ctl_expr	exist next state
EF ctl_expr	exists finally
AG ctl_expr	forall globally
AX ctl_expr	forall next state
AF ctl_expr	forall finally
E [ctl_expr U ctl_expr]	exists until
A [ctl_expr U ctl_expr]	forall until

- **AX** p is true in a state s if for all states s' where there is a transition from s to s' , p is true in s' [36].
- **EF** p is true in a state s^0 if there exists a series of transitions $s_0 \rightarrow s_1, s_1 \rightarrow s_2, \dots, s_{n-1} \rightarrow s_n$ such that p is true in s_n [36].
- **AF** p is true in a state s_0 if for all series of transitions $s_0 \rightarrow s_1, s_1 \rightarrow s_2, \dots, s_{n-1} \rightarrow s_n$ p is true in s_n [36].
- **EG** p is true in a state s_0 if there exists an infinite series of transitions $s_0 \rightarrow s_1, s_1 \rightarrow s_2, \dots$ such that p is true in every s_i [36].
- **AG** p is true in a state s_0 if for all infinite series of transitions $s_0 \rightarrow s_1, s_1 \rightarrow s_2, \dots$ p is true in every s_i [36].
- **E[p U q]** is true in a state s_0 if there exists a series of transitions $s_0 \rightarrow s_1, s_1 \rightarrow s_2, \dots, s_{n-1} \rightarrow s_n$ such that p is true in every state from s_0 to s_{n-1} and q is true in state s_n [36].
- **A[p U q]** is true in a state s_0 if for all series of transitions $s_0 \rightarrow s_1, s_1 \rightarrow s_2, \dots, s_{n-1} \rightarrow s_n$ p is true in every state from s_0 to s_{n-1} and q is true[36].

An Example CTL formulae

- SPEC RF out = 2

It is possible to reach a state in which out = 2

- SPEC EF ($\text{Started} \wedge \neg \text{Ready}$)

If possible to get a state where “Started” holds but “Ready” does not hold.

- SPEC AG (AF DeviceEnabled)

“DeviceEnabled” is always true somewhere along every path starting anywhere: i.e. “DeviceEnabled” holds infinitely often along every path.

- SPEC AG (EF Restart)

From any state it is possible to get to a state for which “Restart” holds.

5.3.2 Linear temporal logic in NuSMV

NuSMV model checker supports the expressions which is designed with this logic. There exist an infinite state sequence in this logic. The each state in this sequence has a different successor state. NuSMV supports the LTL syntax as described in the Table 5.2.

Semantics of LTL operators is as follows [37]:

- $X p$ is true at time t if p is true at time $t + 1$ [37].
- $F p$ is true at time t if p is true at some time $t_0 \geq t$ [37].
- $G p$ is true at time t if p is true at all times $t_0 \geq t$ [37].
- $p U q$ is true at time t if q is true at some time $t_0 \geq t$, and for all time t_{00} (such that $t \leq t_{00} < t_0$) p is true [37].
- $p V q$ is true at time t if q holds at all time steps $t_0 \geq t$ up to and including the time step t_{00} where p also holds. Alternatively, it may be the case that p never holds in which case q must hold in all time steps $t_0 \geq t$ [37].
- $Y p$ is true at time $t > t_0$ if p holds at time $t - 1$. $Y p$ is false at time t_0 [37].
- $Z p$ is equivalent to $Y p$ with the exception that the expression is true at time t_0 [37].
- $H p$ is true at time t if p holds in all previous time steps $t_0 \leq t$ [37].

- $O p$ is true at time t if p held in at least one of the previous time steps $t_0 \leq t$ [37].
- $p S q$ is true at time t if q held at time $t' \leq t$ and p holds in all time steps t'' such that $t_0 < t'' \leq t$ [37].
- $p T q$ is true at time t if p held at time $t' \leq t$ and q holds in all time steps t'' such that $t' \leq t'' \leq t$. Alternatively, if p has never been true, then q must hold in all time steps t'' such that $t' \leq t'' \leq t$ [37].

Table 5.2: The syntax of LTL formulas recognized by NuSMV [37]

Simple Expression	A Simple Boolean Expression
$! \text{ ltl_expr}$	logical not
$ \text{ ltl_expr} \& \text{ ltl_expr}$	logical and
$ \text{ ltl_expr} \text{ ltl_expr}$	logical or
$ \text{ ltl_expr} \text{ xor } \text{ ltl_expr}$	logical exclusive or
$ \text{ ltl_expr} \text{ xnor } \text{ ltl_expr}$	logical NOT exclusive or
$ \text{ ltl_expr} \rightarrow \text{ ltl_expr}$	logical implies
$ \text{ ltl_expr} \leftrightarrow \text{ ltl_expr}$	logical equivalence
$ X \text{ ltl_expr}$	next state
$ G \text{ ltl_expr}$	globally
$ F \text{ ltl_expr}$	finally
$ \text{ ltl_expr} U \text{ ltl_expr}$	until
$ \text{ ltl_expr} V \text{ ltl_expr}$	releases
$ Y \text{ ltl_expr}$	previousstate
$ Z \text{ ltl_expr}$	not previous state not
$ H \text{ ltl_expr}$	historically
$ O \text{ ltl_expr}$	once
$ \text{ ltl_expr} S \text{ ltl_expr}$	since
$ \text{ ltl_expr} T \text{ ltl_expr}$	triggered

6. MODELING TV SOFTWARE USING SMV

In this thesis, the audio parts of the embedded TV software are modelled using SMV language in order to check properties for the model. A non-deterministic user agent is modeled in order to realize the human user. This non-deterministic user-agent is designed to have an ability to click any remote controller (RC) button at any time.

Even though this decision results in an almost full coverage of user inputs it also causes state space explosion problem during the verification process due to the large number of possibilities present in the model.

6.1 Audio Source Control Module of TV

Some audio parts of the embedded TV software is modeled in SMV language in order to check the properties on the model.

- In the model, nine input audio sources are designed:
 - HDMI (High Definition Multimedia Interface) input source: This input source is commonly used for the quality issues. Because audio/video (AV) signal is transmitted as digital.
 - SCART input source: This input source is the mostly used input source for the decades. It is an analog source.
 - Component (YPbPr) input source: It is an another analog source.
 - DVB-C (CABLE) input source: It stands for “Digital Video Broadcasting – Cable”. It is a digital cable transmission system developed by the DVB Project.
 - Three DVB-C Channel are modeled for this input source.
 - DVB-T (TERRESTRIAL) input source: It is a technical standard, developed by the DVB project. In this source, TV audio/video (AV) signal is transmitted by radio waves. It has become the mostly used

format for digital broadcasting in the world since it was first published in 1997.

- Three DVB-T Channel are modelled for this input source.
- DVB-S (SATELLITE): It is the DVB standard for the satellite. Satellite TV is a system of transmission of audio and video signals relayed from the communication satellites.
 - Three DVB-S Channel are modeled for this input source.
- USB Input source: This source is commonly used to play the Audio/Video (AV) contents over USB flash disks.
- ATV Input source: This is the analog TV input source.
- PC input source: This input source was designed to get AV from the PCs. However, newly PCs have HDMI output.
- Various menu items that appear on the user audio menu of the TV software are modeled. These items contain:
 - Sound Mode items that used to enhance the sound according to the user selection,
 - Natural Sound Mode
 - Speech Sound Mode
 - Music Sound Mode
 - User Sound Mode
 - Movie Sound Mode
 - Standard Sound Mode
 - Night Mode On/Off selection that is used to minimize the sound level,
 - Bluetooth On/Off selection that is used to provide sound transmission over Bluetooth,
 - TV Location item that is used to calibrate the sound performance according to the TV location,

- Sound Type items Stereo, Dual1, Dual2, AVL On/Off item that is used to keep the loudness level of the audio while the sudden loudness changes according to the broadcasts,
- HP Volume item that is used to control the audio level of the connected Headphone,
- TV Speaker On/Off item that is used to enable or to disable the Speaker of the TV,
- ARC (Audio Return Channel) On/Off item,
- Scart Volume item that is used to control the volume level of the Scart output of the TV,
- Optic Out item that is used to adjust the Optical audio output of the TV and
- RestoreToDefault item is used to reset the audio parameters.
- The Items for enabling/disabling the audio paths, audio amplifier and audio decoder are also modeled for this model.
- Additionally the audio codecs are desired to be included in the model. But It expand the model as excessive. Thus audio codecs are not included.

6.2 NuSMV Module of Audio Source Control Module

SMV language is used to model audio related embedded TV Software. The implemented model is checked in a powerful 12-core server system with 48GB of memory. NuSMV [7] model checker, which has an ability to give information about the diameter of the BDD, is used to check the properties on the implemented model. The diameter information of the BDD and the total required time for model checking process, which are needed for the experiment results, are provided from NuSMV model checker.

At the appendix section, an example part of the audio source control model is presented as a state transition diagram. Additionally, the source code of the model in SMV language is presented at appendix section.

The complete source code contains 25 modules and 1500 lines. The modules are explained below:

- Main module:

Main module exists between the first and 286th lines of the code in the Appendix B. This module includes all the declarations (TV buttons, TV input sources, source menu items, TV audio initialization related check variables, UI menu states, sound menu items, sound modes, DTV channels ..etc), child module definitions (started at line 165), all initializations (started at line 231) and the properties to be checked (started at line 275).

- “InitTVAudio” module:

This module exists between 286th and 321th lines of the code in the Appendix B. “IsTVAudioInitd” state is set, “AudioDecoders” state is set, “DigitalAudioAmplifier” state is set and “SetAudioPaths” state is set as “run” in this module. This module stands for the initialization of Audio related items.

- “MenuClick” module:

This module exists between 323th and 378th lines of the code in the Appendix B. This module is used to enable or disable the Sound menu status of the model. If the value of “UImenuStates” is “UserAudioMenuUI”, then it means that the UI Sound Menu is printed on the TV screen. If the value of “UImenuStates” is none, it means no UI menu item is printed on the screen of the TV. This module is used to determine if the Sound menu should be printed on the screen or not.

- “SetUITVLocation” module:

This module exists between 379th and 410th lines of the code in the Appendix B. This module is used to switch the TV location state of the model according to the left (mvk_left) and right (mvk_right) buttons of the RC. There exist two possible values: “Stand” and “WallMounted”. TV Location is an item of the SoundMenu of the TV. Thus to alter the values of the TV location, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_TVLocation previously

(Sound menu was printed on the screen and the TV Location item should be selected previously).

- “SetUIOpticOut” module:

This module exists between 413th and 445th lines of the code in the Appendix B. This module is used to switch the UIOpticOut state of the model according to the left (mvk_left) and right (mvk_right) buttons of the RC. There exist two possible values: “AUTO” and “PCM”. Optic Out is an item of the SoundMenu of the TV. Thus to change the values of the UIOpticOut, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_OpticOut previously (Sound menu was printed on the screen and the Optic out item should be selected previously).

- “SetRestoreToDefault” module:

This module exists between 448th and 565th lines of the code in the Appendix B. This module is used to reset the values of all the sound menu items according to the ok (mvk_ok) button of the RC. The values of the states UIBluetooth, UINightMode, UITVLocation, UISoundType, UIAVL, UIARC,UITV_Speaker, UIOpticOut, UI_HPVolume, UI_ScartVolume and UISoundMode are set to OFF, OFF, Stand, stereo, FALSE, FALSE, FALSE, PCM, 4, 4 and SoundMode_natural in orderly. SetRestoreToDefault is an item of the SoundMenu of the TV. Thus to reset all the values of the sound menu, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_RestoreToDefault previously (Sound menu was printed on the screen and the RestoreToDefault item should be selected previously).

- “SetUIHPVol” module:

This module exists between 568th and 598th lines of the code in the Appendix B. This module is used to switch the UI_HPVolume state of the model according to the left (mvk_left) and right (mvk_right) buttons of the RC. There exist five possible values: 0,1,2,3 and 4. UI_HPVolume is an item of the SoundMenu of the TV. Thus to change the values of the UI_HPVolume, UImenustates state should have been UserAudioMenuUI and the

UISoundMenu state should have been SoundMenu_HPVolume previously (Sound menu was printed on the screen and the headphone volume item should be selected previously). Actually, this module is designed to change the value of UI_HPVolume between 0 and 100. However, the value range is too big and a hundred volume step is not necessary for the model. Thus the number of steps for the HP volume are decreased to 5 steps (0,1,2,3,4).

- “SetUIScartVol” module:

This module exists between 600th and 631th lines of the code in the Appendix B. This module is used to switch the UI_ScartVolume state of the model according to the left (mvk_left) and right (mvk_right) buttons of the RC. There exist five possible values: 0,1,2,3 and 4. UI_ScartVolume is an item of the SoundMenu of the TV. Thus to change the values of the UI_ScartVolume, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_ScartVolume previously (Sound menu was printed on the screen and the scart volume item should be selected previously). Actually, this module is designed to change the value of UI_ScartVolume between 0 and 100. Nevertheless, the value range is too big and a hundred volume steps are not necessary for the model. Thus the number of steps for the Scart volume are decreased to 5 steps (0,1,2,3,4).

- “SetUISoundType” module:

This module exists between 634th and 692th lines of the code in the Appendix B. This module is used to change the value of the UI_SoundType state. There exist four possible values: stereo (initial and default value), mono, dual1 and dual2. UI_SoundType is an item of the SoundMenu of the TV. Thus to change the values of the UI_SoundType, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_soundType previously (Sound menu was printed on the screen and the sound type item should be selected previously).

- “SetUIAVL” module

This module exists between 694th and 720th lines of the code in the Appendix B. This module is used to change the value of the UIAVL state. There exist

two possible values: TRUE or FALSE. UIAVL is an item of the SoundMenu of the TV. Thus to change the values of the UIAVL, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_avl previously (Sound menu was printed on the screen and the avl item should be selected previously). Additionally, UIAVL is switchable with left (mvk_left) or right (mvk_right) button of the RC, if the UISoundMode is SoundMode_standart or SoundMode_user.

- “SetUITV_Speaker” module:

This module exists between 724th and 748th lines of the code in the Appendix B. This module is used to change the value of the UITV_Speaker state. There exist two possible values: TRUE or FALSE. UITV_Speaker is an item of the SoundMenu of the TV. Thus to change the values of the UITV_Speaker, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_TVSpeaker previously (Sound menu was printed on the screen and the tv speaker item should be selected previously).

- “SetUIARC” module:

This module exists between 749th and 773th lines of the code in the Appendix B. This module is used to change the value of the UIARC state. There exist two possible values: TRUE or FALSE. UIARC is an item of the SoundMenu of the TV. Thus to change the values of the UIARC, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_ARC previously (Sound menu was printed on the screen and the arc item should be selected previously).

- “SetUINightMode” module

This module exists between 775th and 809th lines of the code in the Appendix B. This module is used to change the value of the UINightMode state. There exist two possible values: ON or OFF. UINightMode is an item of the SoundMenu of the TV. Thus to change the values of the UINightMode, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_nightMode previously (Sound menu was printed on the screen and the nightmode item should be

selected previously). Additionally, UINightMode is switchable with left (mvk_left) or right (mvk_right) button of the RC, only if the UISoundMode is neither SoundMode_standart nor SoundMode_user.

- “SetUIBluetooth” module

This module exists between 810th and 842th lines of the code in the Appendix B. This module is used to change the value of the UIBluetoothstate. There exist two possible values: ON or OFF. UIBluetoothis an item of the SoundMenu of the TV. Thus to change the values of the UIBluetooth, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_bluetooth previously(Sound menu was printed on the screen and the bluetooth item should be selected previously).

- “SetUISoundMode” module:

This module exists between 845th and 985th line of the code in the Appendix B. This module is used to set the value of UISoundMode state via left (mvk_left) and right (mvk_right) buttons of the RC. There exist six possible UISoundMode values: SoundMode_music, SoundMode_standart, SoundMode_user, SoundMode_movie, SoundMode_natural and SoundMode_speech. UISoundModeis an item of the SoundMenu of the TV. Thus to change the values of the UISoundMode, UImenustates state should have been UserAudioMenuUI and the UISoundMenu state should have been SoundMenu_soundMode previously (Sound menu was printed on the screen and the SoundMode item should be selected previously).

- “SoundMenu” module:

This module exists between 987th and 1176th line of the code in the Appendix B. This module is used to set the value of UISoundMenu state via left (mvk_left) and right (mvk_right) buttons. This state represents the sound menu items of the TV. Thus to change the values of the UISoundMenu, UImenustates state should have been UserAudioMenuUI previously (Sound menu was printed on the screen previously).

- “changeDVB_T_Program” module:

This module exists between 1178th and 1210th line of the code in the Appendix B. This module is used to set the value of `dvb_t_channel` state via channel plus (`mvk_channelPlus`) and channel minus (`mvk_channelMinus`) of the RC. This state represents the tuning operation onto a dvb-t channel in Terrestrial input source. Thus to change the values of the `dvb_t_channel` state, `UImenuStates` state should have been none and `TV_InputSource` state should have been `InputSource_TERRESTRIAL` (There must not be any UI printed on the screen of the TV and the input source was selected as Terrestrial previously).

- “changeDVB_C_Program” module:

This module exists between 1214th and 1240th line of the code in the Appendix B. This module is used to set the value of `dvb_c_channel` state via channel plus (`mvk_channelPlus`) and channel minus (`mvk_channelMinus`) of the RC. This state represents the tuning operation onto a dvb-c channel in Cable input source. Thus to change the values of the `dvb_c_channel` state, `UImenuStates` state should have been none and `TV_InputSource` state should have been `InputSource_CABLE` (There must not be any UI printed on the screen of the TV and the input source was selected as cable previously).

- “changeDVB_S_Program” module:

This module exists between 1246th and 1282th line of the code in the Appendix B. This module is used to set the value of `dvb_s_channel` state via channel plus (`mvk_channelPlus`) and channel minus (`mvk_channelMinus`) of the RC. This state represents the tuning operation onto a dvb-s channel in Satellite input source. Thus to change the values of the `dvb_s_channel` state, `UImenuStates` state should have been none and `TV_InputSource` state should have been `InputSource_SATELLITE` (There must not be any UI printed on the screen of the TV and the input source was selected as satellite previously).

- “userClick” module:

This module exists between 1286th and 1370th line of the code in the Appendix B. This module is used to represent the nondeterministic user RC button clicks. When a button is clicked, `rcButton` state is set as

nondeterministically and isClicked state is set as TRUE. When isClicked is set as TRUE other modules handle the job related with the rcButton state. After that, they set isClicked to FALSE to allow this module to set rcButton again nondeterministically.

- “eliminateUnnecessaryUserClicks” module:

This module exists between 1373th and 1395th lines of the code in the Appendix B. There exist so many RC buttons defined in this module however; some of them are related with this model. Because, this model includes not all the features of the TV, it includes just audio related parts. By the way, this model can be developed for the future works. Thus, the irrelevant RC button clicks are cleared in this module by setting isClicked to FALSE.

- “source_click” module:

This module exists between 1397th and 1463th lines of the code in the Appendix B. This module is used to enable or disable the UI source menu status of the model. If the value of UImenuStates is SourceMenuUI, then it means that the UI Source Menu is printed on the TV screen. If the value of “UImenuStates” is none, it means no UI menu item is printed on the screen of the TV. This module is used to determine if the source menu should be printed on the screen or not.

- “source_change” module:

This module exists between 1469th and 1600th lines of the code in the Appendix B. This module is used to set the value of SourceMenuItems state via left (mvk_left) and right (mvk_right) buttons. This state represents the source menu items of the TV. Thus to change the values of the SourceMenuItems state, UImenustates state should have setSourceMenuUI previously via source_click module (Source menu was printed on the screen previously).

- “source_change_monitor” module:

This module exists between 1604th and 1696th lines of the code in the Appendix B. This module is used to set the value of TV_InputSource. The

user navigates between the SourceMenuItems by source_change module via left (mvk_left) and right (mvk_right) buttons. In source_change_monitor module, the user selects the input source via ok (mvk_ok) button.

If the user selects terrestrial input source (if the user click ok button while the state value of SourceMenuItems is SourceMenu_TERRESTRIAL), then the value of dvb_t_channel state is set as dvb_t_ch_one.

If the user selects cable input source (if the user click ok button while the state value of SourceMenuItems is SourceMenu_CABLE), then the value of dvb_c_channel state is set as dvb_c_ch_one.

If the user selects terrestrial input source (if the user click ok button while the state value of SourceMenuItems is SourceMenu_SATELLITE), then the value of dvb_s_channel state is set as dvb_s_ch_one.

- “Volume” module:

This module exists between 1699th and 1786th lines of the code in the Appendix B. This module is used to change the value of TVVolumeLevel via volume up (mvk_volumePlus) and volume down (mvk_volumeMinus) RC buttons. This module is also used to change the value of the isTVMute status of the TV via mute (mvk_mute) RC Button. SoundExist state values are decided in this module according to the isTVMute state, TVVolumeLevel and other relevant states.

The state diagram of the “source_change” module (exists between 1470th and 1600th line of the code in Appendix B) is represented in Figure 6.1. The navigation between the values of the SourceMenuItems state is provided by the left (mvk_left) and right (mvk_right) RC buttons.

The state diagram of the dvb-t part of the “source_change_monitor” module (exists between 1603th and 1697th line of the code in Appendix B) is represented in Figure 6.2. In this module, If the value of the isClicked is TRUE and the value of rcButton is mvk_ok while the value of UImenuStates is SourceMenuUI and the value of SourceMenuItems is SourceMenu_TERRESTRIAL, then the value of TV_InputSource turns to TERRESTRIAL (check the assignment statements at the line 1608).

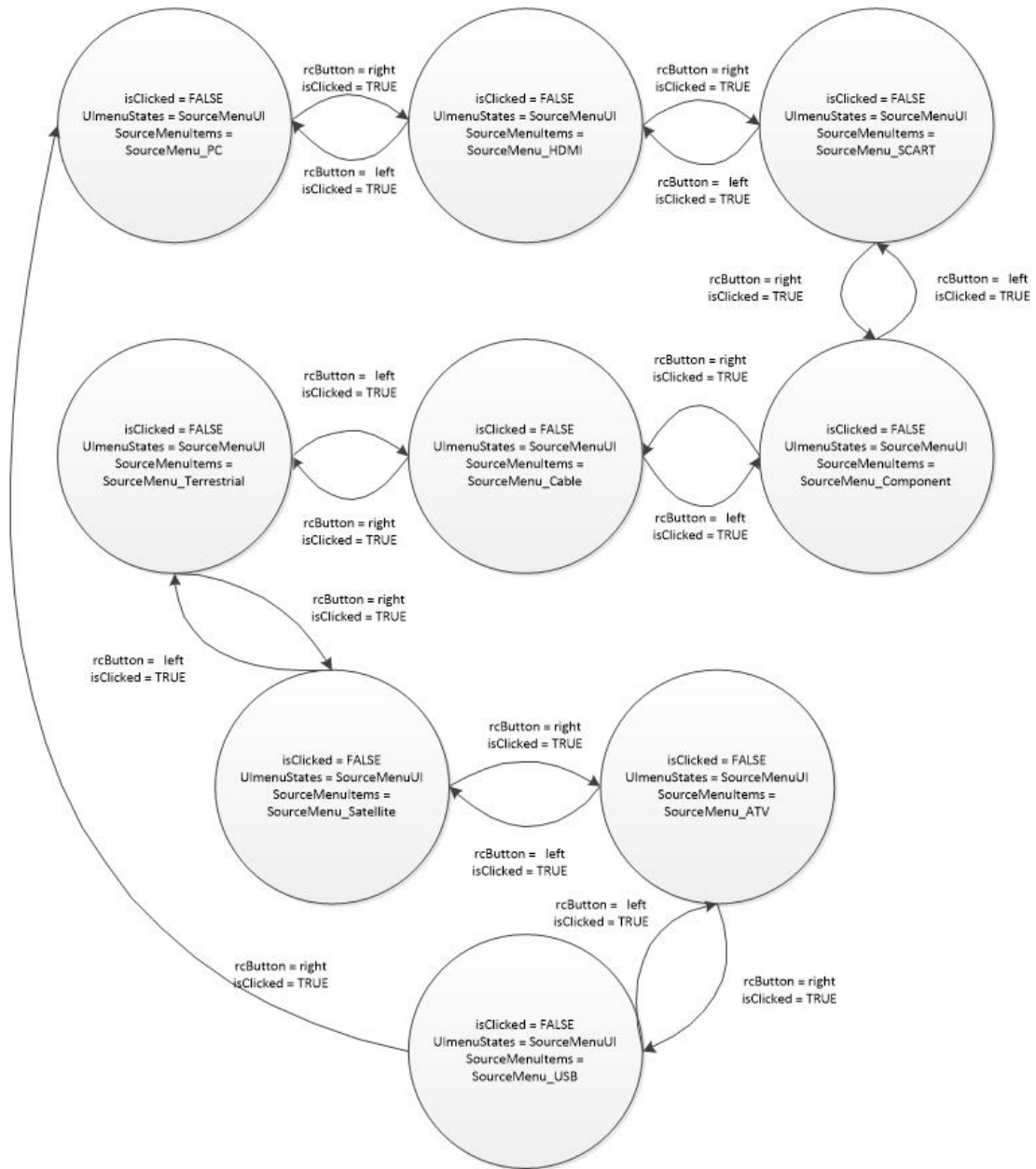


Figure 6.1 : State diagram of the source menu navigation

IsTVAudioInitd should also be TRUE for this purpose. But It is not specified in the Figure 6.2. because of preventing complex view of the figure.

The navigation between the values of the dvb_t_channel state is provided by the channel up (mvk_channelPlus) and channel down (mvk_channelMinus) RC buttons. This navigation is designed in changeDVB_T_Program module (exists between 1178th and 1210th line of the code in Appendix B).

The state diagram of the dvb-c part of the “source_change_monitor” module (exists between 1603th and 1697th line of the code in Appendix B) is represented in Figure 6.3. In this module, If the value of the isClicked is

TRUE and the value of rcButton is mvk_ok while the value of UImenuStates is SourceMenuUI and the value of SourceMenuItems is SourceMenu_CABLE, then the value of TV_InputSource turns to CABLE (check the assignment statements at the line 1608).

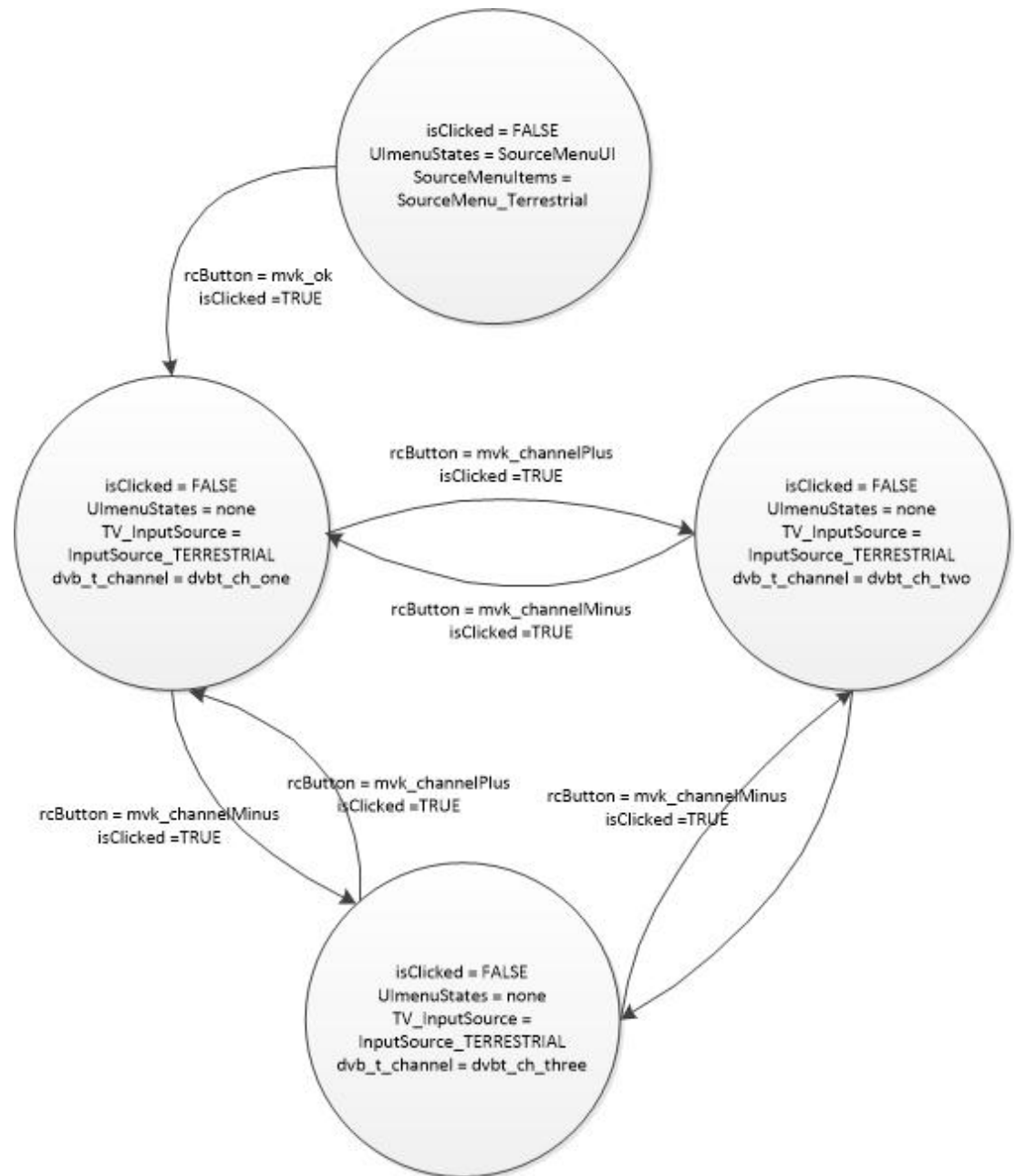


Figure 6.2 : State diagram of the dvb-t channel navigation

IsTVAudioInitd should also be TRUE for this purpose. But It is not specified in the Figure 6.3. because of preventing complex view of the figure.

The navigation between the values of the dvb_t_channel state is provided by the channel up (mvk_channelPlus) and channel down (mvk_channelMinus)

RC buttons. This navigation is designed in changeDVB_C_Program module (exists between 1213th and 1245th line of the code in Appendix B).

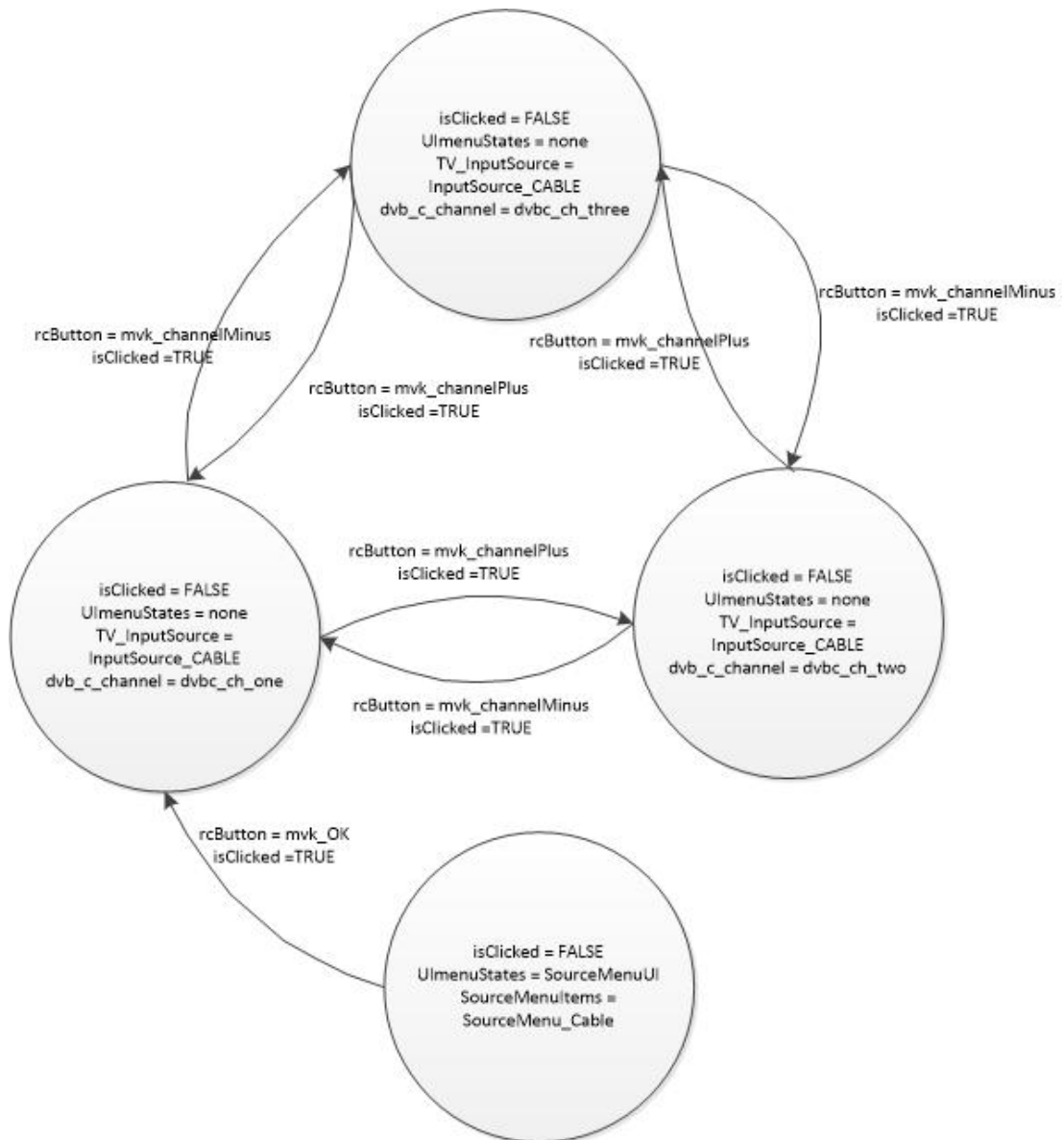


Figure 6.3 : State diagram of the dvb-c channel navigation

The state diagram of the dvb-s part of the “source_change_monitor” module (exists between 1603th and 1697th line of the code in Appendix B) is represented in Figure 6.4. In this module, If the value of the isClicked is TRUE and the value of rcButton is mvk_ok while the value of UImenuStates is SourceMenuUI and the value of SourceMenuItems is SourceMenu_SATELLITE, then the value of TV_InputSource turns to SATELLITE (check the assignment statements at the line 1608).

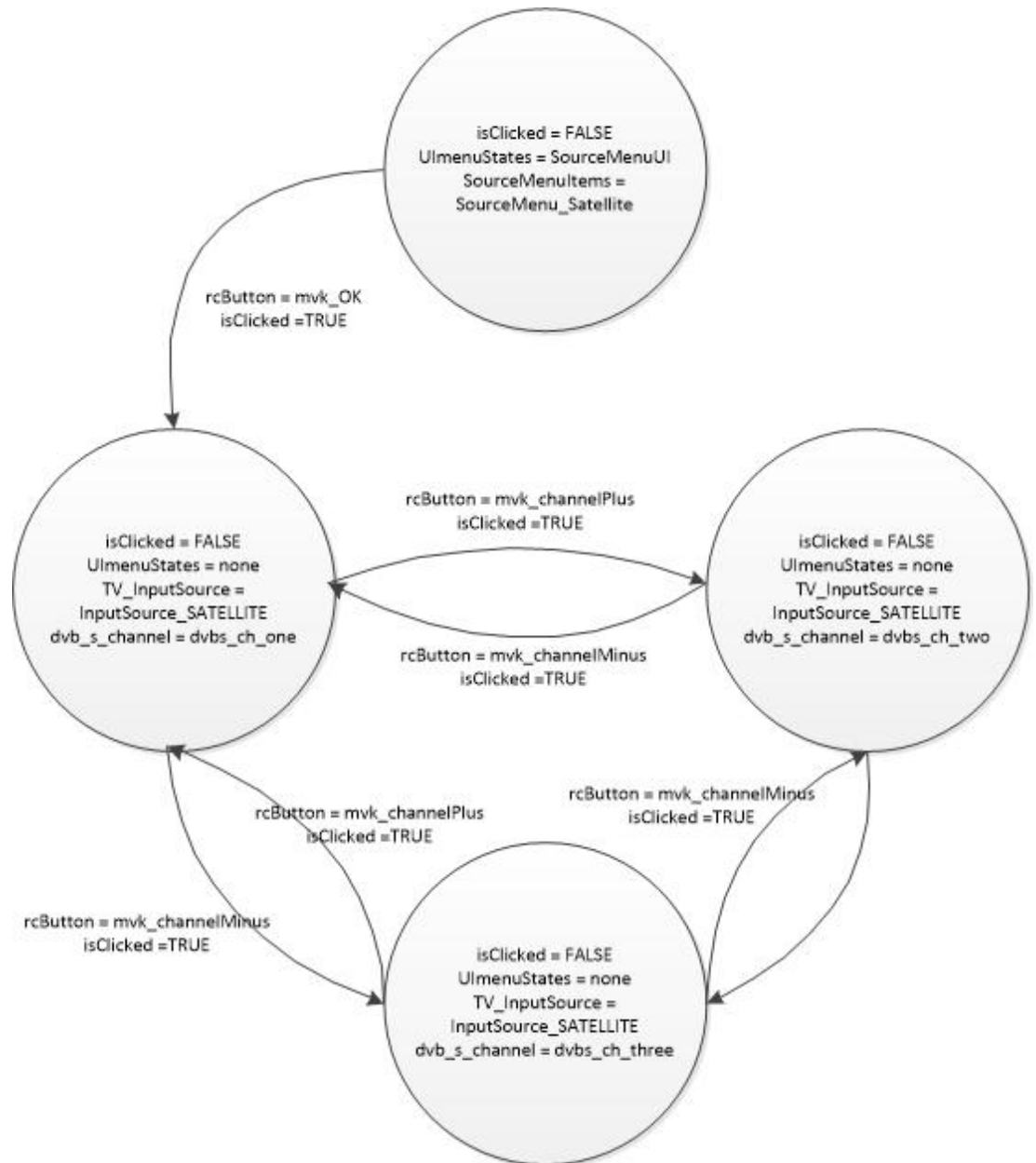


Figure 6.4 : State diagram of the dvb-s channel navigation

IsTVAudioInited should also be TRUE for this purpose. But It is not specified in the Figure 6.4. because of preventing complex view of the figure.

The navigation between the values of the dvb_t_channel state is provided by the channel up (`mvk_channelPlus`) and channel down (`mvk_channelMinus`) RC buttons. This navigation is designed in `changeDVB_S_Program` module (exists between 1247th and 1282th line of the code in Appendix B).

6.3 CTL Specifications To be Checked

In this thesis, a total of four specifications (2 Liveness and 2 Safety) are checked on the TV Model. The first (L1) and the second (L2) Specifications presented in the Table 6.1 are Liveness specifications and the third (S1) and the fourth (S2) specifications are Safety specifications.

Table 6.1 : Specifications

Types	Specifications
L1	AG(isClicked=TRUE ->EF isClicked=FALSE)
L2	EF(isTVMute=FALSE)
S1	AF(TVVolumeLevel!=21)
S2	AG(SoundExist=TRUE -> AF (IsTVAudioInitied!=FALSE))

When an RC button is clicked by the user-agent, “isClicked” state is set as true to provide the related process to be able to do its work according to the related RC keystroke. While “isClicked” state is true, no other RC button is clicked by the user-agent. After the related work done by the related process according to the keystroke, the “isClicked” shall be set as false to permit the user-agent to click another RC button. Because of that L1 is used for a liveness property in the model.

“TVMute” state controls the audio mute status of the TV. TVMute always must not be true for the users to hear audio. Because of that L2 is used for another liveness property.

Audio volume level of the TV is modeled in 21 levels. Because of that, the audio volume level must not exceed 21. Thus S1 is used for a safety property.

To be able to hear audio from the television, audio related HW items (audio decoders, digital audio amplifier etc...) should have been enabled. “IsTVAudioInitied” state stands for the initialization of the audio related HW items. Because of that S2 is used for a safety property. My main purpose is not to check only these specifications (in Table 6.1.) exactly, but also check other specifications according to the Embedded Software model. I checked these specifications (in Table 6.1.) just for an example for an embedded TV model.

7. TV LOGS

TV logs include the remote controller (RC) keystroke histories recorded during the manual testing of the TV by a human tester.

Table 7.1 : Log keys and corresponding RC buttons

Log Keys	RC Buttons
KERNEL received key: 1	power,
KERNEL received key: 2	number,
KERNEL received key: 3	FAV,
KERNEL received key: 4	3D,
KERNEL received key: 5	RED,
KERNEL received key: 6	GREEN,
KERNEL received key: 7	YELLOW,
KERNEL received key: 8	BLUE,
KERNEL received key: 9	info,
KERNEL received key: a	exit,
KERNEL received key: b	menu,
KERNEL received key: c	back,
KERNEL received key: d	tools,
KERNEL received key: e	guide,
KERNEL received key: f	volumePlus,
KERNEL received key: 10	volumeMinus,
KERNEL received key: 11	channelPlus,
KERNEL received key: 12	channelMinus,
KERNEL received key: 13	sourceChange,
KERNEL received key: 14	mute,
KERNEL received key: 15	play,
KERNEL received key: 16	stop,
KERNEL received key: 17	pause,
KERNEL received key: 18	backward,
KERNEL received key: 19	forward,
KERNEL received key: 1a	previous,
KERNEL received key: 1b	next,
KERNEL received key: 1c	audio,
KERNEL received key: 1d	subtitle,
KERNEL received key: 1e	pvr
KERNEL received key: 1f	up,
KERNEL received key: 20	down,
KERNEL received key: 21	left,
KERNEL received key: 22	right,
KERNEL received key: 23	ok

At the development phase of the embedded TV software, manual testers use RC to test the TV, which has the embedded software that is being implemented.

The TV logs are processed to produce new user agents according to the approaches that are presented in this thesis.

There exist so many TV brands in the market. Thus there exist so many type of RCs. Since there may exist significant differences among the RC logs of various TV models and various TV brands, a template RC is created and used to unify this variety in this thesis. While creating the template, most common (mainly used) RC buttons are considered as a main priority. Four TV brands (Grundig TV RC, Vestel TV RC, Samsung TV RC and LG TV RC) are benchmarked for this purpose. According to the benchmark and the ease of use in the implemented model, 35 RC buttons are designed in this template. This template is used for the approaches represented in this thesis. Additionally, the log keys are assigned according to the each RC buttons at the template. The log keys and the corresponding RC buttons are listed in Table 7.1.

The RC keystroke histories include the any of the log keys in Table 7.1. These keystroke histories are the input components for user-agent creator approaches in this thesis.

8. TEST LOG BASED SPECIFICATION OF USER BEHAVIOR

A non-deterministic user agent implementation causes state space explosion problem in the model checking process. The following three approaches are proposed in order to overcome this problem.

8.1 Direct Log Based Specification

The user agent is produced directly from the logs of the TV in this approach. The log file is read as an input and transformed to a user agent module in SMV language directly without any modifications.

State Transition Systems (STS) are produced by each transformation with representing each user click as a separate transition between states as depicted in Figure 8.1.

The transformation process consists of two steps:

- I. Create states that has a number of RC Keys plus one in the log file.
- II. Assign the RC Keys to the transitions between the previous and next states.

An example application for “Direct log based Specification” approach is presented to explain the approach in more detail. RC log file number one (Table 8.1) is used for this purpose. The corresponding RC Buttons are presented in Table 8.2 and the corresponding RC Key transition system of the output user agent module is presented in Figure 8.2.

The RC Key transition system, which is presented in Figure 8.2, is the result of the Direct Log Based Specification approach. In this transition system, each corresponding RC button are modeled as transitions. The results of this approach is explained in the result section.

The RC keys of the example log file no: 1 are presented in orderly in Table 8.1.

The Corresponding RC buttons are presented according to the RC keys in Table 8.2 .

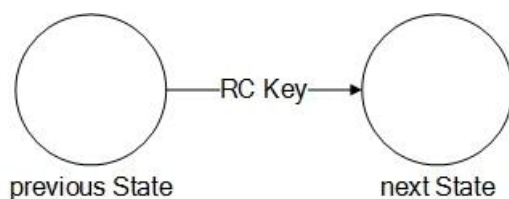


Figure 8.1 : RC Key (transition) is assigned just between the previous and the next states

Table 8.1 : Example log content

Log File No: 1	
	KERNEL received key: 14
	KERNEL received key: f
	KERNEL received key: f
	KERNEL received key: 13
	KERNEL received key: 22
	KERNEL received key: 22
	KERNEL received key: 22
	KERNEL received key: 22
	KERNEL received key: 22
	KERNEL received key: 23
	KERNEL received key: 11
	KERNEL received key: 11
	KERNEL received key: 11
	KERNEL received key: 12
	KERNEL received key: b

Table 8.2 : RC keys and corresponding RC buttons

RC Keys	Corresponding RC Button
KERNEL received key: 14	Mute
KERNEL received key: f	volumePlus
KERNEL received key: f	volumePlus
KERNEL received key: 13	sourceChange
KERNEL received key: 22	right
KERNEL received key: 22	right
KERNEL received key: 22	right
KERNEL received key: 22	right
KERNEL received key: 22	right
KERNEL received key: 23	ok
KERNEL received key: 11	channelPlus
KERNEL received key: 11	channelPlus
KERNEL received key: 11	channelPlus
KERNEL received key: 12	channelMinus
KERNEL received key: b	menu

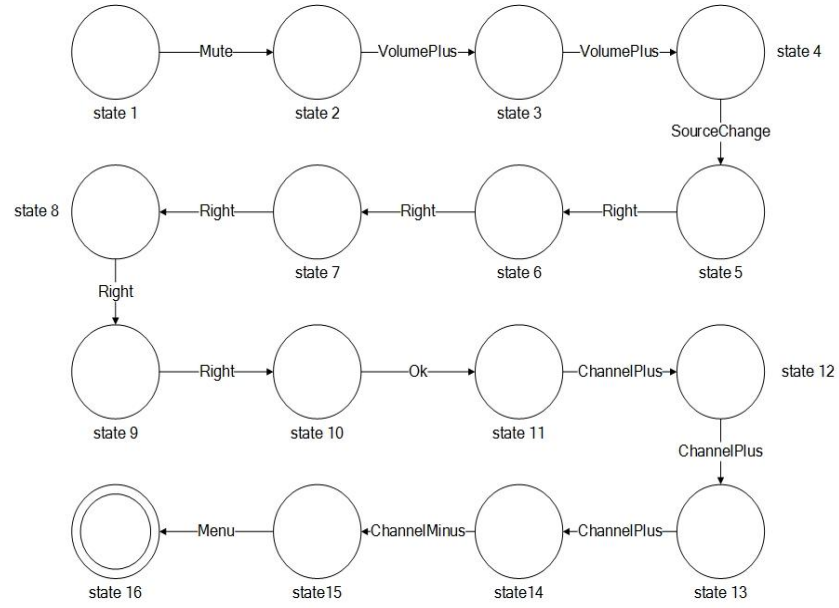


Figure 8.2 : The corresponding RC Key transition system of the output user agent module

8.2 Injecting Nondeterminism to Log Based Specification

Direct log based specification approach in the previous section produces a totally deterministic user agent module for the model checking process. However, a fully deterministic user behavior is closer to a manual test by human than rather than performing the real intention to perform model checking process. It is much more appropriate to generate some kind of a user module that contains an insufficient amount of non-determinism that can eliminate the state space problems resulted by a fully non-deterministic user.

The previous approach is improved by injecting non-determinism into the user agent of the model produced by the method introduced in the former section. In this approach, number of clicks (frequencies) are calculated for each button of the RC which exists in the examined log. Each different button is transformed to transitions between the states. If the frequency of the button is larger than one, then this button also produces a non-deterministic self-transition to the emerging state as depicted in Figure 8.3. The transformation process consists of the two steps as depicted below:

- I. Calculate the frequencies of the each separate RC Key on the log and construct a frequency table.

- II. Implement a user agent STS by using the frequency table of the log.
 - a. Create $n+1$ states for n distinct RC Key types present in the log. Since each RC key corresponds to a transition we need $n+1$ states.
 - b. Assign the RC Keys to the transitions between the states by applying the rules below:
 - i. If the frequency of the key is equal to one, this key (transition) is assigned just between the previous and the next states of the log file (see Figure 8.1).
 - ii. If the frequency of the key is greater than one, this key (transition) is assigned between the next state and the previous state and this key (application) is also be assigned as a loop transition in the same state (see Figure 8.3).

An example application for “Injecting non-determinism to log based specification” approach is presented to explain the approach in more detail. For this purpose, log file number one (Table 8.1) is used.

Initially the frequencies of each keystroke type are calculated in the log. The resulted frequency table is presented in Table 8.3 and the corresponding RC Buttons according to the RC Keys in Table 8.3 is presented in Table 8.4.

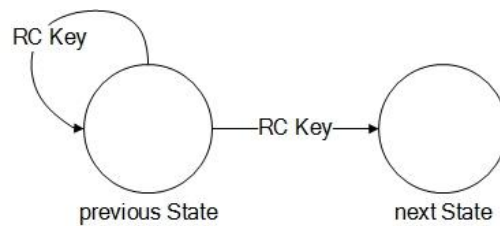


Figure 8.3 : RC Key (transition) is assigned between the previous and the next states and this RC key (transition) is also be assigned as a loop transition in the same state to provide a non-deterministic transition.

The frequency table for the “Log-file-number-1” in (Table 8.1) is presented in Table 8.3.

The Corresponding RC buttons are presented in Table 8.4. according to the RC keys in Table 8.3.

Table 8.3 : RC keys and corresponding keystroke frequencies

RC Keys	Keystroke Frequencies
KERNEL received key: 14	1
KERNEL received key: f	2
KERNEL received key: 13	1
KERNEL received key: 22	5
KERNEL received key: 23	1
KERNEL received key: 11	3
KERNEL received key: 12	1
KERNEL received key: b	1

Table 8.4 : RC keys and corresponding RC buttons

RC Keys	Corresponding RC Button
KERNEL received key: 14	mute
KERNEL received key: f	volumePlus
KERNEL received key: 13	sourceChange
KERNEL received key: 22	right
KERNEL received key: 23	ok
KERNEL received key: 11	channelPlus
KERNEL received key: 12	channelMinus
KERNEL received key: b	menu

Next, a user agent STS is implemented by using the frequency table of the log. An example STS constructed from the log in Table 8.1 and frequency table in Table 8.3 is in Figure 8.4. The RC Key transition system is presented in Figure 8.4 is the result of the Injecting Non-determinism to Log Based Specifications approach. In this transition system, each corresponding distinct RC button are transformed to transitions. The results of this approach can be found in result section.

8.3 Combining Log-Based Specification

The approach in the previous section produces a non-deterministic user agent module for the model checking process. In order to implement different types of user agents, even those that is not directly present in the test logs, the STS's, which are reproduced in the previous approaches, can be combined. The transformation process consists of two steps:

- I. Calculate the frequencies of the each distinct RC Key on the log and construct a frequency table.

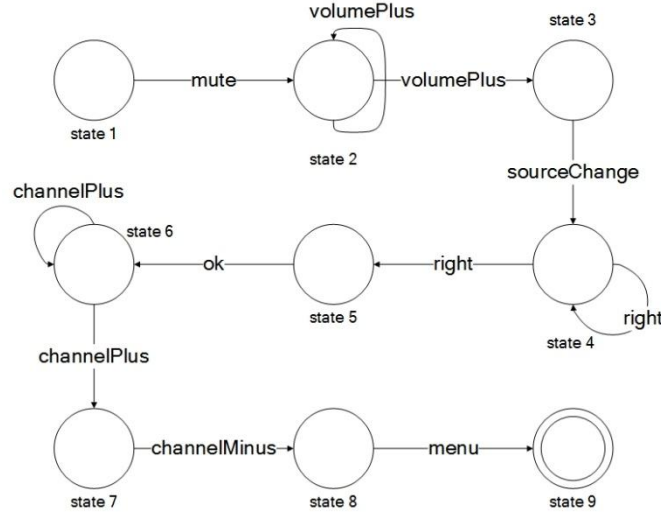


Figure 8.4 : Generated transition system for “Injecting Non-determinism to Log Based Specifications” approach for the log file on the Table 8.1.

- II. Implement a combined user-agent STS from two different log files, namely log1 and log2. Let’s assume that number of distinct key types in log1 is larger than number of distinct key types in log2.
 - a. Create $m+1$ states for m distinct RC Key types present in the log of log1.
 - b. Create n states for n distinct RC Key types present in the log of log2.
 - c. If the number of distinct key types are equal for both of the logs produce $m+1$ and $n+1$ states for each log respectively.
 - d. Assign the RC Keys to the transitions between the states by applying the rules below:
 - i. If the frequency of the key is equal to one, this key (transition) is assigned just between the previous and the next states of the log file. (see Fig. 8.1).
 - ii. If the frequency of the key is greater than one, this key (transition) is assigned between the next state and the previous state and this key (application) is also be assigned as a loop transition in the same state. (see Fig. 8.3).
 - iii. Regarding the order of key stroke orders in each STS we couple the states in the same order (if exists) and connect them

with transitions using corresponding key strokes in the log file.
(see Fig. 8.5).

An example application for “Combining Log Based Specification” approach is presented to explain the approach in more detail. For this purpose, log1 (Table 8.1) and log2 (Table 8.5) are used.

Table 8.5 : Example log content

Log File No: 2
KERNEL received key: b
KERNEL received key: 20
KERNEL received key: 20
KERNEL received key: 20
KERNEL received key: 20
KERNEL received key: 20
KERNEL received key: 20
KERNEL received key: 20
KERNEL received key: 20
KERNEL received key: 22
KERNEL received key: 22
KERNEL received key: 22
KERNEL received key: 22
KERNEL received key: 21
KERNEL received key: 21
KERNEL received key: 21
KERNEL received key: 22
KERNEL received key: 22

The RC keys of the example log file no: 2 are presented in orderly.

Table 8.6 : RC keys and corresponding frequencies

RC Keys	Corresponding RC Button
KERNEL received key: b	1
KERNEL received key: 20	8
KERNEL received key: 22	6
KERNEL received key: 21	3

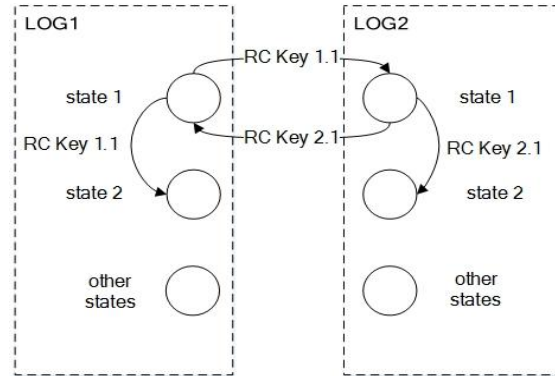
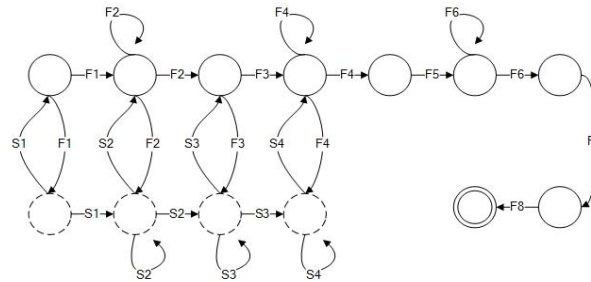
The frequency table for the “Log-file-number-2” in (Table 8.5).

Next the user-agent STS is created by using the frequency tables of each logs and combining the STS of log1 and log2. The constructed user-agent STS is shown in Figure 8.6.

The Corresponding RC buttons are presented according to the RC keys in Table 8.7.

Table 8.7 : RC keys and corresponding RC buttons

RC Keys	Corresponding RC Button
KERNEL received key: b	Menu
KERNEL received key: 20	down
KERNEL received key: 22	right
KERNEL received key: 21	left

**Figure 8.5 : Each RC Keys in the logs are assigned as a transition between the corresponding states of the other log file****Figure 8.6 : Generated transition system for “Combining Log Based Specifications” approach for the log files on the Table 8.1 and Table 8.5.**

In Fig. 8.6, the states with straight lines and the states with dotted lines are formed respectively according to the example log1 (see Table 8.1) and the example log2 (see Table 8.5).

The constructed STS depicted in Figure 8.6 includes some loop transitions formed to inject non-determinism to the user-agent as a result of the frequency value of the corresponding RC Key in the related log files in Table 8.3 and Table 8.6. Additionally, the states of the STS1 and STS2 are coupled with the related transitions (RC Buttons). Thus the non-determinism level of the created user-agent is increased.

Different combination approaches can be applied to increase the non-determinism level of the user-agent and decrease the state space and the query process time of the

total model. A straight approach is used to increase the non-deterministic level of the user-agent and decrease the state space and the query process time of the total model. The results of this approach are explained in result section. Table 8.8 indicates that the corresponding RC Buttons according to the transitions in the Figure 8.6.

Table 8.8 : The RC buttons which corresponds to the transitions in Figure 8.6

Transitions	Corresponding RC Button
F1	Mute
F2	Volume Plus
F3	Source Change
F4	Right
F5	Ok
F6	Channel Plus
F7	Channel Minus
F8	Menu
S1	Menu
S2	Down
S3	Right
S4	Left

9. RESULTS

In this thesis, three approaches are proposed to reduce the process time that is required to complete the model checking and shrink the radius of the Binary Decision Diagram (BDD), which is used by NuSMV to represent the state space during the model checking process of the part of embedded TV software. Ten different TV logs are used to apply the proposed approaches and a total of four safety and liveness properties are checked in order to compare the results of the proposed approaches.

The proposed approaches in this thesis are applied to the specified log files and the user-agent modules created by the approaches are used to check the specified properties for the implemented TV audio control software model.

The BDD diameter of the original model is 242 and the time required to model check the original model is 3158881 milliseconds (approx. 53 minutes). The process time that is required to complete the model checking is measured by the outputs of the “real” part of the “time” command of the linux OS.

Tables 9.1, 9.2 and 9.3 represent the results according to the each corresponding log files. The properties depicted in Table 6.1. are checked for the original model for the each corresponding log files. Afterwards the properties are checked for the models, which include the newly created user agents according to the proposed approaches. The results of these properties are all TRUE for both the original model and the models that include the created user agent models.

The user-agent modules are edited to shrink the state space of the model (the BDD radius of the model) in the proposed methods. Since the used scenarios include specific RC key sequences, it is important to consider that the properties that are checked should be in accordance with the input scenarios. Even though this can be seen as the main disadvantage of our approach, model checking generally aims towards safety and liveness properties at a broader context like checking for

deadlocks. It is paid attention to use such liveness and safety properties in the experiments.

Table 9.1 : Direct log based approach results

Used Log File	Diameter of the BDD	Process Time of the Model (ms)
Log File no 1	33	520
Log File no 2	61	402
Log File no 3	67	410
Log File no 4	33	421
Log File no 5	94	810
Log File no 6	24	352
Log File no 7	31	437
Log File no 8	22	317
Log File no 9	44	383
Log File no 10	18	316

Table 9.1 presents resulted diameter of the BDD and the process times that are required to complete the model checking process for the Direct Log Based Specification approach according to the specified log files. The Diameters of the BDDs and the process times are strictly decreased according to the results of the original model.

Table 9.2 presents resulted diameter of the BDD and the process times that are required to complete the model checking process for the “Injecting Non-determinism to Log Based Specifications” approach according to the specified log files. The Diameters of the BDDs and the process times are significantly decreased according to the results of the original model.

Table 9.2 : Injecting Non-determinism to log based approach results

Used Log File	Diameter of the BDD	Process Time of the Model (ms)
Log File no 1	55	466
Log File no 2	37	347
Log File no 3	37	344
Log File no 4	24	382
Log File no 5	60	464
Log File no 6	34	341
Log File no 7	55	410
Log File no 8	33	293
Log File no 9	20	354
Log File no 10	13	302

Table 9.3. presents resulted diameter of the BDD and the process times that are required to complete the model checking process for the “Log Based Specifications” approach according to the specified log files. The diameters of the BDDs are strictly decreased except for the Combination of 5 and 8. The BDD diameter of this combination is larger than the diameter of the BDD of the original model (which is 242). However the process time of this combination is very short compared with the original models process time (which is 3158881 msec). Even if the diameter of the model including user-agent combination of 5 and 8 is increased, the process time is still very short compared with the original model.

Table 9.3 : Combining log based approach results

Used Log File	Diameter of the BDD	Process Time of the Model (ms)
Combination of 1 and 2	75	1008
Combination of 4 and 3	68	513
Combination of 5 and 8	254	120870
Combination of 6 and 3	147	24870
Combination of 7 and 8	108	1014
Combination of 5 and 10	71	609
Combination of 9 and 2	66	484
Combination of 9 and 3	66	457

10. CONCLUSION

The study in this thesis proposes a technique that includes three approaches for preventing the state space explosion problem that is specific to the embedded TV software verification.

It is able to shrink the time that is required for verification significantly compared to using a fully non-deterministic user-agent module in the implemented embedded TV software model by injecting non-determinism to enhance the test scenarios used in legacy test suites. The proposed approaches can be applied with any kind of model checker by implementing an appropriate transformation programs.

Another contribution that is presented in this thesis is combining multiple test scenarios to form a new non-deterministic scenario, altering between two test scenarios.

Different combination approaches and even combination operators may also be implemented to obtain more detailed user-agent modules. The future research direction is to further automate the combination approach and examine the outcomes of using different combinational operators.

REFERENCES

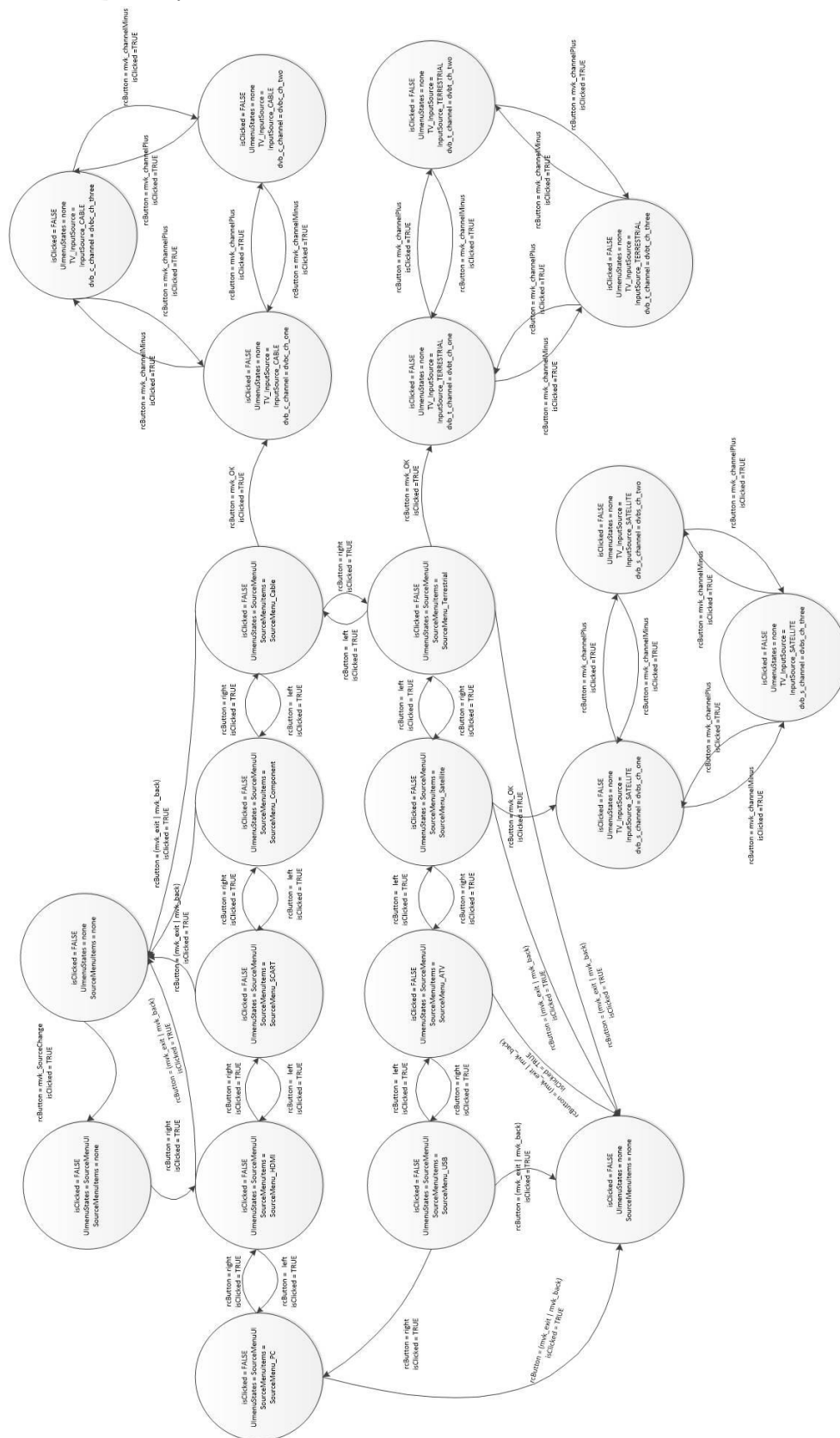
- [1] **Takatori, S., Matsumoto, S., Saiki, S., and Nakamura, M.**(2014). “Considering value-added services using smart TV in cloud-enabled home network system,” in *Proc. IEEE 3rd Global Conference on Consumer Electronics*, Tokyo, Japan, pp. 268-270, Oct.
- [2] **Perakakis, E. and Ghinea, G.**(2015). “A proposed model for cross-platform web 3D applications on smart TV systems,”in *Proc. ACM 20th International Conference on 3D Web Technology*, Crete, Greece, pp. 165-166, June.
- [3] **Breitman, K., Endler, M., Pereira, R. and Azambuja, M.**(2010). “When TV Dies, Will It Go to the Cloud?” *Computer*, vol.43, no.4, pp.81-83, April.
- [4] **Clarke, E., Emerson, E. and Sifakis, J.**(2009). “Turing lecture: model checking—algorithmic verification and debugging,” *Communications of the ACM*, vol.52, no.11, pp.78,84, Nov.
- [5] **Baier, C. and Katoen, J. P.**(2008). *Principles of model checking*, Vol. 26202649. Cambridge: MIT press.
- [6] **Valmari, A.**(1998). “The state explosion problem,”*Lectures on Petri nets I: Basic models*, Springer Berlin Heidelberg, pp. 429-528.
- [7] **Cimatti, A., Clarke, E., Giunchiglia, F. and Roveri, M.**(1999). “NuSMV: A new symbolic model verifier,” *Computer Aided Verification*, Springer Berlin Heidelberg, pp. 495-499.
- [8] **Demri, S., Laroussinie, F. and Schnoebelen, P.**(2006). “A parametric analysis of the state explosion problem in model checking,”*Journal of Computer and System Sciences*, vol.72, no.4, pp.547-575, June.
- [9] **Silva, V. D., Kroening, D. and Weissenbacher, G.**(2008). “A survey of automated techniques for formal software verification,”*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.27, no.7, pp.1165-1178, July.
- [10] **Jensen, K., Kristensen, L. M. and Mailund, T.**(2012). “The sweep-line state space exploration method,”*Theoretical Computer Science*, vol.429, pp.169-179, April.
- [11] **Androutsopoulos, K., Clark, D., Harman, M., Krinke, J. and Tratt, L.**(2013). “State-based model slicing: A survey,”*ACM Computing Surveys*, vol.45, no.4, Article No. 53, Aug.
- [12] **Silva, J.**(2012). “A vocabulary of program slicing-based techniques,”*ACM Computing Surveys*, vol.44, no.3, Article No. 12, June

- [13] **Holzmann, G. J., Joshi, R. and Groce, A.**(2010). "Swarm verification techniques,"*IEEE Transactions on Software Engineering*, vol.37, no.6, pp.845-857, Dec.
- [14] **Bellettini, C., Camilli, M., Capra, L. and Monga, M.**(2014). "Symbolic state space exploration of RT systems in the cloud,"in *Proc. IEEE 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Timisoara, Romania, pp. 295-302, Sept.
- [15] **Pekovic, V., Zlokolica, V., Zloh, J., Katona, M. and Teslic, N.**(2012). "Automated System for Testing and Verification of Control Access Kernel Functionality in Set-Top Boxes," in *Proc. IEEE International Conference on Consumer Electronics*, Las Vegas, USA, pp. 25-26, Jan.
- [16] **Markande, P., Joshi, P. V. and Manvi, S. S.**(2013). "Enhancing Effectiveness and Reducing Time of System Validation for Interactive Multimedia LED TV," in *Proc. IEEE International Conference on Advances in Computing, Communications and Informatics*, Mysore, India, pp. 1738-1743, Aug.
- [17] **Kastelan, I., Pekovic, V., Zlokolica, V., Zloh, J. and Trifunovic, D.**(2012). "Simultaneous Automated Verification of Conditional Access System on Multiple TV Sets," in *Proc. IEEE International Conference on Consumer Electronics*, Berlin, Germany, pp. 269-270, Sept.
- [18] **Kovacevic, M., Kovacevic, B., Pekovic, V. and Stefanovic, D.**(2014). "Framework for Automatic Testing of Set-Top Boxes," in *Proc. IEEE 22nd Telecommunications Forum*, Belgrade, Serbia, pp. 1091-1094, Nov.
- [19] **Marinkovic, V., Kordic, B., Popovicand, M. and Pekovic, V.**(2013). "A Method for Creating the Operational Profile of TV/STB Device to be Used for Statistical Testing," in *Proc. IEEE International Conference on Computer as a Tool*, Zagreb, Croatia, pp. 93-97, July.
- [20] **Murphy, C., Zoomkawalla, Z. and Narita, K.**(2013). "Automatic Test Case Generation and Test Suite Reduction for Closed-Loop Controller Software," Technical Reports of Dept. Of Computer and Information Science, University of Pennsylvania Philadelphia, PA 19104, Rep. 979, Feb.
- [21] **Kukolj, S., Marinkovic, V. and Popovic, M.**(2013). "Selection and Prioritization of Test Cases by Combining White-Box and Black-Box Testing Methods," in *Proc. IEEE 3rd Eastern European Regional Conference on the Engineering of Computer Based Systems*, Budapest, Hungary, pp. 153-156, Aug.
- [22] **Tekcan, T., Zlokolica, V., Pekovic, V., Teslic, N. and Gündüzalp, M.**(2012). "User-driven automatic test-case generation for DTV/STB reliable functional verification," *IEEE Transactions on Consumer Electronics*, vol. 58, no.2, pp 587-595, May 2012.
- [23] **Marinkovic, V., Kastelan, I., Vranic, N., Marceta, Z. and Pekovic, V.**(2012). "Protocol Independent Remote Controller

Emulator for automated testing of Multiple DTV/STB,” in *Proc. IEEE 20th Telecommunications Forum*, Belgrade, Serbia, pp. 1064-1067, Nov.

- [24] **Cordeiro, L., Fischer, B. and Marques-Silva, J.**(2012). “SMT-Based Bounded Model Checking for Embedded ANSI-C Software,” *IEEE Transactions on Software Engineering*, vol.38, no.4, pp. 957-974, July-Aug.
- [25] **Giannakopoulou, D. and Magee, J.**(2013). “Fluent Model Checking for Event-based Systems,” *ACM SIGSOFT Software Engineering Notes*, vol.28, no.5, pp. 257-266, Sept.
- [26] **Gnesi, S., Latella, D. and Massink, M.**(1999). “Model Checking UML Statechart Diagrams using JACK,” in *Proc. IEEE 4th International Symposium on High-Assurance Systems Engineering*, Washington DC, USA, pp. 46-55, Nov.
- [27] **Ji, W., Wei, D. and Zhi-Chang, Q.**(2002). “Slicing Hierarchical Automata for Model Checking UML Statecharts,” *Formal Methods and Software Engineering*, Springer Berlin Heidelberg, vol.2495. pp. 435-446, Oct.
- [28] **Leiter, E. and Heaven, W.**(2013). “Principles of Model Checking” in *Proc. IEEE 35th International Conference on Software Engineering*, San Francisco, USA, pp. 592-601, May.
- [29] **Kripke, S. A.**(1962). “Semantical considerations on modal logic” In *Proc. A colloquium on Modal and Many-Valued Logics*, Helsinki.
- [30] **Clarke, E., Grumberg, O. and Hamaguchi, K.**(1997). “Another Look at LTL Model Checking. Formal Methods in System Design” *Formal Methods in System Design*, 10(1):57--71, February.
- [31] **Baier, C. and Katoen, J-P.**(2008). “Principals of Model Checking” *The MIT Press*, 55 Hayward Street, Cambridge, MA, pp. 8.
- [32] **Cimatti, A., Clarke, E., Giunchiglia, F. and Roveri, M.**(2000). “Nusmv: a new symbolic model checker,” *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 2, no. 4, pp. 410–425,
- [33] **Andersen, H. R.**(1996). “An Introduction to Binary Decision Diagrams.” *Online*, September.
- [34] **Schneider, K.**(2013). “Verification of Reactive Systems: Formal Methods and Algorithm” *Springer Science & Business Media*. pp.45.
- [35] **Url-1**<http://nusmv.fbk.eu/NuSMV/userman/v11/html/nusmv_25.html>
- [36] **Cavada, R., Cimatti, A., Jochim, C. A., Keighren, G., Olivetti, E., Pistore, M., Roveri, M. and Tchaltev, A.**(2010). “NuSMV 2.5 User Manual” *FBK-irst – Via Sommarive 18, 38055 Provo Italy*. pp.37.
- [37] **Cavada, R., Cimatti, A., Jochim, C. A., Keighren, G., Olivetti, E., Pistore, M., Roveri, M. and Tchaltev, A.**(2010). “NuSMV 2.5 User Manual” *FBK-irst – Via Sommarive 18, 38055 Provo Italy*. pp.38.

APPENDIX A:



APPENDIX B:

SourceCode.smv

```
1  MODULE main
2      VAR
3      --RC Buttons are defined below.
4          rcButton: {
5              mvk_none,
6              mvk_power,
7              mvk_number,
8
9              mvk_FAV,
10             mvk_3D,
11             mvk_RED,
12             mvk_GREEN,
13             mvk_YELLOW,
14             mvk_BLUE,
15             mvk_info,
16             mvk_exit,
17             mvk_menu,
18             mvk_back,
19             mvk_tools,
20             mvk_guide,
21             mvk_volumePlus,
22             mvk_volumeMinus,
23             mvk_channelPlus,
24             mvk_channelMinus,
25             mvk_sourceChange,
26             mvk_mute,
27             mvk_play,
28             mvk_stop,
29             mvk_pause,
30             mvk_backward,
31             mvk_forward,
32             mvk_previous,
33             mvk_next,
34             mvk_audio,
35             mvk_subtitle,
36             mvk_pvr,
37             mvk_up,
38             mvk_down,
39             mvk_left,
40             mvk_right,
41             mvk_ok
42
43         };
44
45         --USER RC CLICKS
46         --isClicked is defined to control the completion of the
47         related jobs. If isClicked is false, userclick process can click
48         another button. If isClicked is true, the button had been clicked
49         and there may be some works to do according to the related rcButton.
50         isClicked: boolean;
51
52         --AUDIO CODEC DEFINES
53         dtvAudioCodec
54         : {none, DTV_AUD_MPEG, DTV_AUD_AC3, DTV_AUD_HEAAC, DTV_AUD_DP};
55         usbAudioCodec
56         : {none, USB_AUD_MPEG, USB_AUD_AC3, USB_AUD_HEAAC, USB_AUD_DP};
```

```

57
58
59         --TV INPUT SOURCES
60         TV_InputSource:{none,InputSource_HDMI,
61 InputSource_SCART, InputSource_COMPONENT, InputSource_CABLE,
62 InputSource_TERRESTRIAL,
63 InputSource_SATELLITE,InputSource_ATV,InputSource_USB,InputSource_PC
64 };
65         SourceMenuItems:{none,SourceMenu_HDMI, SourceMenu_SCART,
66 SourceMenu_COMPONENT, SourceMenu_CABLE, SourceMenu_TERRESTRIAL,
67 SourceMenu_SATELLITE,SourceMenu_ATV,SourceMenu_USB,SourceMenu_PC};
68
69         IsTVAudioInitied:boolean;          --INIT TV AUDIO
70
71         --JOBS
72         SetAudioPaths: {set,not_set};          -- specifies if
73 the audio paths are set or not
74         DigitalAudioAmplifier: {run,not_run};          --
75 specifies if the audio paths are set or not
76         AudioDecoders:{set,not_set};          -- specifies if the
77 audio paths are set or not
78
79         --UI STATES
80         --UI States specifies which menu is activated.
81         UImenuStates: { none,
82                               UserAudioMenuUI,
83                               SourceMenuUI
84 };
85
86         --UINightMode
87         UINightMode: {ON,OFF};          --Night Mode is
88 generally used in TVs to check the audio level at night.
89
90         --UIBluetooth
91         UIBluetooth: {ON,OFF};          -- Specifies if a
92 Bluetooth audio transmission is applied or not
93
94         --UITVLocation
95         UITVLocation: {WallMounted,Stand};          -- TV Location
96 item is generally used in TVs to apply sound enhancement parameters
97 according to the TV location.
98         UISoundType : {stereo,dual1,dual2};          -- The
99 known sound types are defined.
100        UIAVL: boolean;          --Auto Volume
101 Levelling
102        UIARC: boolean;          --Audio Return
103 Channel
104        UITV_Speaker :boolean;          --Speaker is ON of
105 OFF
106        UIOpticOut: {AUTO,PCM};          -- Specifies the
107 audio optic output behaviour of the TV.
108        UI_HPVolume: 0..21;          -- just 21 level is
109 defined for Headphone volume. more levels cause more BDD diameters.
110        UI_ScartVolume: 0..21;          -- just 21 level is
111 defined for Scart volume. more levels cause more BDD diameters.
112        userClickOrder: 0..1000;          -- This value
113 keeps the number of keystrokes of RC. This is used the user-agents
114 that is created by our approaches. This is not used for totally
115 non_deterministic user-agent.
116
117         --USER SOUND MENU ITEMS

```

```

118         UISoundMenu: {      SoundMenu_none,--wii be developed
119                             SoundMenu_soundMode,
120                             SoundMenu_nightMode,
121                             SoundMenu_bluetooth,
122                             --SoundMenu_selectBTDevice,    --
123 will be defined on future work
124                             SoundMenu_TVLocation,
125                             SoundMenu_soundType,
126                             SoundMenu_graphEq,
127                             SoundMenu_avl,
128                             SoundMenu_HPvolume,
129                             SoundMenu_TVSpeaker,
130                             SoundMenu_ARC,
131                             --SoundMenu_AudioOut            --
132 will be defined on future work
133                             --SoundMenu_ScartOutput,
134         -- will be defined on future work
135                             SoundMenu_ScartVolume,
136                             SoundMenu_OpticOut,
137                             SoundMenu_RestoretoDefault
138         };
139         --SoundMode
140         UISoundMode: {
141             SoundMode_user,
142             SoundMode_standart,
143             SoundMode_movie,
144             SoundMode_natural,
145             SoundMode_speech,
146             SoundMode_music
147         };
148
149         --TV Channels
150         -- Total 9 TV Channel are defined.
151
152         dvb_t_channel:{none,dvbt_ch_one,dvbt_ch_two,dvbt_ch_three};
153
154         dvb_c_channel:{none,dvbc_ch_one,dvbc_ch_two,dvbc_ch_three};
155
156         dvb_s_channel:{none,dvbs_ch_one,dvbs_ch_two,dvbs_ch_three};
157
158         --TV Volume
159         SoundExist: boolean;
160         isTVMute:boolean;
161         TVVolumeLevel: 0..21;                                -- just
162 21 level is defined for TV volume. more levels cause more BDD
163 diameters.
164
165         --PROCESSES
166         --
167         proc_MSC : process
168 source_change_monitor(TV_InputSource,isClicked,rcButton,UImenuStates
169 ,SourceMenuItems,IstVAudioInitied,dvb_t_channel,dvb_c_channel,dvb_s_
170 channel);
171
172         --handle The audio initialization of TV.
173         proc_INIT_SOUND : process
174 initTVAudio(IstVAudioInitied,SetAudioPaths,DigitalAudioAmplifier,Aud
175 ioDecoders);
176
177         proc_userCilck : process
178 userClick(isClicked,rcButton,userClickOrder);

```

```

179         proc_TVVolume: process
180     Volume(isClicked,rcButton,UImenuStates,SoundExist,isTVMute,TVVolumeL
181     evel,IstVAudioInitied);
182         proc_menu : process
183     MenuClick(isClicked,rcButton,UImenuStates,UISoundMenu);
184         proc_SoundMenu : process
185     SoundMenu(isClicked,rcButton,UImenuStates,UISoundMenu);
186         proc_UISoundMode : process
187     SetUISoundMode(isClicked,rcButton,UImenuStates,UISoundMenu,UISoundMo
188     de,UIAVL,UINightMode);
189         proc_UIBluetooth : process
190     SetUIBluetooth(isClicked,rcButton,UImenuStates,UISoundMenu,UIBluetoo
191     th);
192         proc_UINightMode : process
193     SetUINightMode(isClicked,rcButton,UImenuStates,UISoundMenu,UINightMo
194     de,UISoundMode);
195         proc_UITVLocation : process
196     SetUITVLocation(isClicked,rcButton,UImenuStates,UISoundMenu,UITVLoca
197     tion);
198         proc_UIAVL : process
199     SetUIAVL(isClicked,rcButton,UImenuStates,UISoundMenu,UIAVL,UISoundMo
200     de);
201         proc_UITVSpeaker : process
202     SetUITV_Speaker(isClicked,rcButton,UImenuStates,UISoundMenu,UITV_Spe
203     aker);
204         proc_UIScartVol : process
205     SetUIScartVol(isClicked,rcButton,UImenuStates,UISoundMenu,UI_ScartVo
206     lume);
207         proc_UIHPVolume : process
208     SetUIHPVol(isClicked,rcButton,UImenuStates,UISoundMenu,UI_HPVolume);
209         proc_UIR2D: process
210     SetRestoreToDefault(isClicked,rcButton,UImenuStates,UISoundMenu,UIBl
211     uetooth,UINightMode,UITVLocation,UISoundType,UIAVL,UIARC,UITV_Speake
212     r,UIOpticOut,UI_HPVolume,UI_ScartVolume,UISoundMode);
213         proc_EliminateClick: process
214     eliminateUnnecessaryUserClicks(isClicked,rcButton,UImenuStates);
215
216         proc_SourceClick: process
217     source_click(isClicked,rcButton,UImenuStates,SourceMenuItems,IstVAud
218     ioInitied);
219         proc_SourceChange: process
220     source_change(isClicked,rcButton,UImenuStates,SourceMenuItems);
221         proc_CC_S: process
222     changeDVB_S_Program(TV_InputSource,UImenuStates,isClicked,rcButton,d
223     vb_s_channel);
224         proc_CC_C: process
225     changeDVB_C_Program(TV_InputSource,UImenuStates,isClicked,rcButton,d
226     vb_c_channel);
227         proc_CC_T: process
228     changeDVB_T_Program(TV_InputSource,UImenuStates,isClicked,rcButton,d
229     vb_t_channel);
230     ASSIGN
231         init (userClickOrder):=1;
232         init (isClicked):=FALSE;
233     --SCALE INITS
234
235         init (dtvAudioCodec) := none;
236         init (usbAudioCodec) := none;
237         init (SoundExist):=FALSE;
238         init (isTVMute):=FALSE;
239         init (TVVolumeLevel):= 2;

```

```

240
241      --TV INIT
242          init(SetAudioPaths):= not_set;
243          init(DigitalAudioAmplifier):=not_run;
244          init(AudioDecoders):=not_set;
245          init(UImenuStates):=none;
246      --menu inits
247          init (UIBluetooth):=OFF;
248          init (UINightMode):=OFF;
249          init (UITVLocation):=Stand;
250          init (UISoundType):=stereo;
251          init (UIAVL):=FALSE;
252          init (UIARC):=FALSE;
253          init (UITV_Speaker):=FALSE;
254          init (UIOpticOut):=PCM;
255          init (UI_HPVolume):=4;
256          init (UI_ScartVolume):=4;
257          init (UISoundMode):=SoundMode_natural;
258
259      --SOURCE CHANGE INITS
260          init (TV_InputSource) := none; -- TV init olunca HDMI'a
261      gececek
262          init (SourceMenuItems):= none;
263
264          init (IsTVAudioInitied):=FALSE;
265
266
267          init(dvb_t_channel):=none;
268          init(dvb_c_channel):=none;
269          init(dvb_s_channel):=none;
270
271
272      FAIRNESS
273          running
274
275      --LIVENESS
276      SPEC AG(isClicked=TRUE ->EF isClicked=FALSE)
277      SPEC EF(isTVMute=FALSE)
278
279      --SAFETY
280      SPEC AF(TVVolumeLevel!=21)
281      SPEC AG(SoundExist=TRUE -> AF (IsTVAudioInitied!=FALSE))
282
283
284      --TV audio initialization parts are defined and will be developed in
285      the future work.
286      MODULE
287      initTVAudio(IsTVAudioInitied,SetAudioPaths,DigitalAudioAmplifier,Aud
288      ioDecoders)
289          ASSIGN
290              next(IsTVAudioInitied):=
291                  case
292                      SetAudioPaths=not_set:TRUE;
293                      TRUE:IsTVAudioInitied;
294                  esac;
295
296              next(AudioDecoders):=
297                  case
298                      SetAudioPaths=not_set:set;
299                      TRUE:AudioDecoders;
300                  esac;

```

```

301
302
303         next (DigitalAudioAmplifier) :=
304         case
305             SetAudioPaths=not_set:run;
306             TRUE:DigitalAudioAmplifier;
307         esac;
308
309
310         next (SetAudioPaths) :=
311         case
312             SetAudioPaths=not_set:set;
313             TRUE:SetAudioPaths;
314         esac;
315
316
317 FAIRNESS
318     running
319
320
321
322 --Enable/Disable the user Sound Menu status
323 MODULE MenuClick(isClicked,rcButton,UImenuStates,UISoundMenu)
324 ASSIGN
325
326     next (UISoundMenu) :=
327     case
328
329
330         (isClicked = TRUE) &
331         (UImenuStates = none) &
332         (rcButton = mvk_menu):SoundMenu_none;
333
334         (isClicked = TRUE) &
335         (UImenuStates = UserAudioMenuUI) &
336         (rcButton = mvk_menu):SoundMenu_none;
337
338         (isClicked = TRUE) &
339         (UImenuStates = UserAudioMenuUI) &
340         (rcButton = mvk_back):SoundMenu_none;
341
342
343
344         TRUE:UISoundMenu;
345
346     esac;
347
348
349     next (UImenuStates) :=
350     case
351         (isClicked = TRUE) &
352         (UImenuStates = none) &
353         (rcButton = mvk_menu):UserAudioMenuUI;
354
355         (isClicked = TRUE) &
356         (UImenuStates = UserAudioMenuUI) &
357         (rcButton = mvk_menu):none;
358
359         (isClicked = TRUE) &
360         (UImenuStates = UserAudioMenuUI) &
361         (rcButton = mvk_back):none;

```

```

362
363         TRUE:UImenuStates;
364
365     esac;
366     next (isClicked):=
367     case
368         (isClicked = TRUE) &(
369             ((UImenuStates = none)&(rcButton = mvk_menu)) |
370             ((UImenuStates = UserAudioMenuUI) & ((rcButton =
371 mvk_menu)|(rcButton = mvk_back))) ) :FALSE;
372         TRUE: isClicked;
373
374     esac;
375 FAIRNESS
376     running
377
378 --Sets the TV Location status.
379 MODULE
380 SetUITVLocation(isClicked,rcButton,UImenuStates,UISoundMenu,UITVLoca
381 tion)
382 ASSIGN
383     next (UITVLocation):=
384     case
385         (rcButton = mvk_left | rcButton = mvk_right) &
386         UImenuStates = UserAudioMenuUI &
387         UISoundMenu = SoundMenu_TVLocation &
388         UITVLocation = WallMounted &
389         isClicked = TRUE : Stand;
390
391         (rcButton = mvk_left | rcButton = mvk_right) &
392         UImenuStates = UserAudioMenuUI &
393         UISoundMenu = SoundMenu_TVLocation &
394         UITVLocation = Stand &
395         isClicked = TRUE : WallMounted;
396
397         TRUE : UITVLocation;
398     esac;
399
400     next (isClicked):=
401     case
402         --(rcButton = mvk_left | rcButton = mvk_right) &
403         UImenuStates = UserAudioMenuUI &
404         UISoundMenu = SoundMenu_TVLocation &
405         isClicked = TRUE: FALSE;
406         TRUE: isClicked;
407     esac;
408 FAIRNESS
409     running
410
411 --Sets the Optic audio output status of the TV
412 MODULE
413 SetUIOpticOut(isClicked,rcButton,UImenuStates,UISoundMenu,UIOpticOut
414 )
415 ASSIGN
416     next (UIOpticOut):=
417     case
418         (rcButton = mvk_left | rcButton = mvk_right) &
419         UImenuStates = UserAudioMenuUI &
420         UISoundMenu = SoundMenu_OpticOut &
421         UIOpticOut = AUTO &
422         isClicked = TRUE : PCM;

```

```

423
424         (rcButton = mvk_left | rcButton = mvk_right) &
425         UImenuStates = UserAudioMenuUI &
426         UISoundMenu = SoundMenu_OpticOut &
427         UIOpticOut = PCM &
428         isClicked = TRUE : AUTO;
429
430         TRUE : UIOpticOut;
431     esac;
432
433     next (isClicked) :=
434     case
435         --(rcButton = mvk_left | rcButton = mvk_right) &
436         UImenuStates = UserAudioMenuUI &
437         UISoundMenu = SoundMenu_OpticOut &
438         isClicked = TRUE: FALSE;
439         TRUE: isClicked;
440     esac;
441
442
443 FAIRNESS
444     running
445
446 --Sets the audio parameters to their initial values.
447 MODULE
448 SetRestoreToDefault(isClicked,rcButton,UImenuStates,UISoundMenu,UIBl
449 uetooth,UINightMode,UITVLocation,UISoundType,UIAVL,UIARC,UITV_Speake
450 r,UIOpticOut,UI_HPVolume,UI_ScartVolume,UISoundMode)
451
452 ASSIGN
453 next (UIBluetooth) :=
454     case
455         (rcButton = mvk_ok) &
456         UImenuStates = UserAudioMenuUI &
457         UISoundMenu = SoundMenu_RestoretoDefault &
458         isClicked = TRUE : OFF;
459         TRUE : UIBluetooth;
460     esac;
461
462 next (UINightMode) :=
463     case
464         (rcButton = mvk_ok) &
465         UImenuStates = UserAudioMenuUI &
466         UISoundMenu = SoundMenu_RestoretoDefault &
467         isClicked = TRUE : OFF;
468         TRUE : UINightMode;
469     esac;
470
471 next (UITVLocation) :=
472     case
473         (rcButton = mvk_ok) &
474         UImenuStates = UserAudioMenuUI &
475         UISoundMenu = SoundMenu_RestoretoDefault &
476         isClicked = TRUE : Stand;
477         TRUE : UITVLocation;
478     esac;
479
480
481 next (UISoundType) :=
482     case
483         (rcButton = mvk_ok) &

```

```

484         UImenuStates = UserAudioMenuUI &
485         UISoundMenu = SoundMenu_RestoretoDefault &
486         isClicked = TRUE : stereo;
487         TRUE : UISoundType;
488     esac;
489
490 next (UIAVL) :=
491     case
492         (rcButton = mvk_ok) &
493         UImenuStates = UserAudioMenuUI &
494         UISoundMenu = SoundMenu_RestoretoDefault &
495         isClicked = TRUE : FALSE;
496         TRUE : UIAVL;
497     esac;
498
499 next (UIARC) :=
500     case
501         (rcButton = mvk_ok) &
502         UImenuStates = UserAudioMenuUI &
503         UISoundMenu = SoundMenu_RestoretoDefault &
504         isClicked = TRUE : FALSE;
505         TRUE : UIARC;
506     esac;
507
508 next (UITV_Speaker) :=
509     case
510         (rcButton = mvk_ok) &
511         UImenuStates = UserAudioMenuUI &
512         UISoundMenu = SoundMenu_RestoretoDefault &
513         isClicked = TRUE : FALSE;
514         TRUE : UITV_Speaker;
515     esac;
516
517 next (UIOpticOut) :=
518     case
519         (rcButton = mvk_ok) &
520         UImenuStates = UserAudioMenuUI &
521         UISoundMenu = SoundMenu_RestoretoDefault &
522         isClicked = TRUE : PCM;
523         TRUE : UIOpticOut;
524     esac;
525
526
527 next (UI_HPVolume) :=
528     case
529         (rcButton = mvk_ok) &
530         UImenuStates = UserAudioMenuUI &
531         UISoundMenu = SoundMenu_RestoretoDefault &
532         isClicked = TRUE : 4;
533         TRUE : UI_HPVolume;
534     esac;
535
536
537 next (UI_ScartVolume) :=
538     case
539         (rcButton = mvk_ok) &
540         UImenuStates = UserAudioMenuUI &
541         UISoundMenu = SoundMenu_RestoretoDefault &
542         isClicked = TRUE : 4;
543         TRUE : UI_ScartVolume;
544     esac;

```

```

545
546 next (UISoundMode) :=
547     case
548         (rcButton = mvk_ok) &
549         UImenuStates = UserAudioMenuUI &
550         UISoundMenu = SoundMenu_RestoretoDefault &
551         isClicked = TRUE : SoundMode_natural;
552         TRUE : UISoundMode;
553     esac;
554
555 next (isClicked) :=
556     case
557         --(rcButton = mvk_ok) &
558         UImenuStates = UserAudioMenuUI &
559         UISoundMenu = SoundMenu_RestoretoDefault &
560         isClicked = TRUE: FALSE;
561         TRUE: isClicked;
562     esac;
563 FAIRNESS
564     running
565
566
567 --Sets the volume level at HP audio output of TV
568 MODULE
569 SetUIHPVol(isClicked,rcButton,UImenuStates,UISoundMenu,UI_HPVolume)
570 ASSIGN
571     next (UI_HPVolume) :=
572         case
573             (rcButton = mvk_right) &
574             UImenuStates = UserAudioMenuUI &
575             UISoundMenu = SoundMenu_HPvolume &
576             UI_HPVolume != 21 &
577             isClicked = TRUE : UI_HPVolume+1;
578
579             (rcButton = mvk_left) &
580             UImenuStates = UserAudioMenuUI &
581             UISoundMenu = SoundMenu_HPvolume &
582             UI_HPVolume != 0 &
583             isClicked = TRUE : UI_HPVolume+(-1);
584
585             TRUE : UI_HPVolume;
586         esac;
587
588     next (isClicked) :=
589         case
590             --(rcButton = mvk_left | rcButton = mvk_right) &
591             UImenuStates = UserAudioMenuUI &
592             UISoundMenu = SoundMenu_HPvolume &
593             isClicked = TRUE: FALSE;
594             TRUE: isClicked;
595         esac;
596 FAIRNESS
597     running
598
599 --Sets the volume level at scart audio output of TV
600 MODULE
601 SetUIScartVol(isClicked,rcButton,UImenuStates,UISoundMenu,UI_ScartVo
602 lume)
603 ASSIGN
604     next (UI_ScartVolume) :=
605         case

```

```

606         (rcButton = mvk_right) &
607         UImenuStates = UserAudioMenuUI &
608         UISoundMenu = SoundMenu_ScartVolume &
609         UI_ScartVolume != 21 &
610         isClicked = TRUE : UI_ScartVolume+1;
611
612         (rcButton = mvk_left) &
613         UImenuStates = UserAudioMenuUI &
614         UISoundMenu = SoundMenu_ScartVolume &
615         UI_ScartVolume != 0 &
616         isClicked = TRUE : UI_ScartVolume+(-1);
617
618         TRUE : UI_ScartVolume;
619     esac;
620
621     next (isClicked):=
622     case
623         --(rcButton = mvk_left | rcButton = mvk_right) &
624         UImenuStates = UserAudioMenuUI &
625         UISoundMenu = SoundMenu_ScartVolume &
626         isClicked = TRUE: FALSE;
627         TRUE: isClicked;
628     esac;
629 FAIRNESS
630     running
631
632
633 --Sets SoundType Status
634 MODULE
635 SetUISoundType(isClicked,rcButton,UImenuStates,UISoundMenu,UISoundTy
636 pe)
637
638 ASSIGN
639     next (UISoundType):=
640     case
641     --STEREO
642         (rcButton = mvk_left) &
643         UImenuStates = UserAudioMenuUI &
644         UISoundMenu = SoundMenu_soundType &
645         UISoundType = stereo &
646         isClicked = TRUE : dual2;
647
648         (rcButton = mvk_right) &
649         UImenuStates = UserAudioMenuUI &
650         UISoundMenu = SoundMenu_soundType &
651         UISoundType = stereo &
652         isClicked = TRUE : dual1;
653     --DUAL1
654         (rcButton = mvk_left) &
655         UImenuStates = UserAudioMenuUI &
656         UISoundMenu = SoundMenu_soundType &
657         UISoundType = dual1 &
658         isClicked = TRUE : stereo;
659
660         (rcButton = mvk_right) &
661         UImenuStates = UserAudioMenuUI &
662         UISoundMenu = SoundMenu_soundType &
663         UISoundType = dual1 &
664         isClicked = TRUE : dual2;
665     --DUAL2
666         (rcButton = mvk_left) &

```

```

667         UImenuStates = UserAudioMenuUI &
668         UISoundMenu = SoundMenu_soundType &
669         UISoundType = dual2 &
670         isClicked = TRUE : dual1;
671
672         (rcButton = mvk_right) &
673         UImenuStates = UserAudioMenuUI &
674         UISoundMenu = SoundMenu_soundType &
675         UISoundType = dual2 &
676         isClicked = TRUE : stereo;
677
678
679         TRUE : UISoundType;
680     esac;
681
682     next (isClicked) :=
683     case
684         --(rcButton = mvk_left | rcButton = mvk_right) &
685         UImenuStates = UserAudioMenuUI &
686         UISoundMenu = SoundMenu_soundType &
687         isClicked = TRUE: FALSE;
688         TRUE: isClicked;
689     esac;
690 FAIRNESS
691     running
692
693 --Set AVL Status
694 MODULE
695 SetUIAVL(isClicked,rcButton,UImenuStates,UISoundMenu,UIAVL,UISoundMo
696 de)
697 ASSIGN
698     next (UIAVL) :=
699     case
700         (UISoundMode=SoundMode_user |
701 UISoundMode=SoundMode_standart)&
702         (rcButton = mvk_left | rcButton = mvk_right) &
703         UImenuStates = UserAudioMenuUI &
704         UISoundMenu = SoundMenu_avl &
705         isClicked = TRUE : !UIAVL;
706
707         TRUE : UIAVL;
708     esac;
709
710     next (isClicked) :=
711     case
712         --(rcButton = mvk_left | rcButton = mvk_right) &
713         UImenuStates = UserAudioMenuUI &
714         UISoundMenu = SoundMenu_avl &
715         isClicked = TRUE: FALSE;
716         TRUE: isClicked;
717     esac;
718
719 FAIRNESS
720     running
721
722
723 -- Set the TV Speaker status.
724 MODULE
725 SetUITV_Speaker(isClicked,rcButton,UImenuStates,UISoundMenu,UITV_Spe
726 aker)
727 ASSIGN

```

```

728     next (UITV_Speaker) :=
729     case
730         (rcButton = mvk_left | rcButton = mvk_right) &
731         UImenuStates = UserAudioMenuUI &
732         UISoundMenu = SoundMenu_TVSpeaker &
733         isClicked = TRUE : !UITV_Speaker;
734
735         TRUE : UITV_Speaker;
736     esac;
737
738     next (isClicked) :=
739     case
740         --(rcButton = mvk_left | rcButton = mvk_right) &
741         UImenuStates = UserAudioMenuUI &
742         UISoundMenu = SoundMenu_TVSpeaker &
743         isClicked = TRUE: FALSE;
744         TRUE: isClicked;
745     esac;
746 FAIRNESS
747     running
748
749 --Set the ARC status
750 MODULE SetUIARC(isClicked,rcButton,UImenuStates,UISoundMenu,UIARC)
751 ASSIGN
752     next (UIARC) :=
753     case
754         (rcButton = mvk_left | rcButton = mvk_right) &
755         UImenuStates = UserAudioMenuUI &
756         UISoundMenu = SoundMenu_ARC &
757         isClicked = TRUE : !UIARC;
758
759         TRUE : UIARC;
760     esac;
761
762     next (isClicked) :=
763     case
764         --(rcButton = mvk_left | rcButton = mvk_right) &
765         UImenuStates = UserAudioMenuUI &
766         UISoundMenu = SoundMenu_ARC &
767         isClicked = TRUE: FALSE;
768         TRUE: isClicked;
769     esac;
770 FAIRNESS
771     running
772
773
774 --Set the Night mode status of TV.
775 MODULE
776 SetUINightMode(isClicked,rcButton,UImenuStates,UISoundMenu,UINightMo
777 de,UISoundMode)
778 ASSIGN
779     next (UINightMode) :=
780     case
781
782         (rcButton = mvk_left | rcButton = mvk_right) &
783         UImenuStates = UserAudioMenuUI &
784         UISoundMenu = SoundMenu_nightMode &
785         UINightMode =ON &
786         isClicked = TRUE : OFF;
787

```

```

788         (UISoundMode != SoundMode_user
789 &UISoundMode!=SoundMode_standart)&
790         (rcButton = mvk_left | rcButton = mvk_right) &
791         UImenuStates = UserAudioMenuUI &
792         UISoundMenu = SoundMenu_nightMode &
793         UINightMode =OFF &
794         isClicked = TRUE : ON;
795
796         TRUE : UINightMode;
797     esac;
798
799     next (isClicked):=
800     case
801         --(rcButton = mvk_left | rcButton = mvk_right) &
802         UImenuStates = UserAudioMenuUI &
803         UISoundMenu = SoundMenu_nightMode &
804         isClicked = TRUE: FALSE;
805         TRUE: isClicked;
806     esac;
807 FAIRNESS
808     running
809
810 --Set the Bluetooth status
811 MODULE
812 SetUIBluetooth(isClicked,rcButton,UImenuStates,UISoundMenu,UIBluetoo
813 th)
814 ASSIGN
815     next (UIBluetooth):=
816     case
817         (rcButton = mvk_left | rcButton = mvk_right) &
818         UImenuStates = UserAudioMenuUI &
819         UISoundMenu = SoundMenu_bluetooth &
820         UIBluetooth = ON &
821         isClicked = TRUE : OFF;
822
823         (rcButton = mvk_left | rcButton = mvk_right) &
824         UImenuStates = UserAudioMenuUI &
825         UISoundMenu = SoundMenu_bluetooth &
826         UIBluetooth = OFF &
827         isClicked = TRUE : ON;
828
829         TRUE : UIBluetooth;
830     esac;
831
832     next (isClicked):=
833     case
834         --(rcButton = mvk_left | rcButton = mvk_right) &
835         UImenuStates = UserAudioMenuUI &
836         UISoundMenu = SoundMenu_bluetooth &
837         isClicked = TRUE: FALSE;
838         TRUE: isClicked;
839     esac;
840 FAIRNESS
841     running
842
843
844 --Navigation between the soundmodes
845 MODULE
846 SetUISoundMode(isClicked,rcButton,UImenuStates,UISoundMenu,UISoundMo
847 de,UIAVL,UINightMode)
848 ASSIGN

```

```

849
850     next (UIAVL) :=
851     case
852         rcButton = mvk_right &
853         UISoundMenu = SoundMenu_soundMode &
854         UImenuStates = UserAudioMenuUI &
855         UISoundMode = SoundMode_standart &
856         isClicked = TRUE : FALSE;
857
858         rcButton = mvk_left &
859         UISoundMenu = SoundMenu_soundMode &
860         UImenuStates = UserAudioMenuUI &
861         UISoundMode = SoundMode_user &
862         isClicked = TRUE : FALSE;
863         TRUE: UIAVL;
864     esac;
865
866     next (UINightMode) :=
867     case
868         rcButton = mvk_left &
869         UISoundMenu = SoundMenu_soundMode &
870         UImenuStates = UserAudioMenuUI &
871         UISoundMode = SoundMode_movie &
872         isClicked = TRUE : OFF;
873
874         rcButton = mvk_right &
875         UISoundMenu = SoundMenu_soundMode &
876         UImenuStates = UserAudioMenuUI &
877         UISoundMode = SoundMode_music&
878         isClicked = TRUE : OFF;
879         TRUE: UINightMode;
880     esac;
881
882     next (UISoundMode) :=
883     case
884
885     --SoundMode_user
886         rcButton = mvk_left &
887         UISoundMenu = SoundMenu_soundMode &
888         UImenuStates = UserAudioMenuUI &
889         UISoundMode = SoundMode_user &
890         isClicked = TRUE : SoundMode_music;
891
892         rcButton = mvk_right &
893         UISoundMenu = SoundMenu_soundMode &
894         UImenuStates = UserAudioMenuUI &
895         UISoundMode = SoundMode_user &
896         isClicked = TRUE : SoundMode_standart;
897
898     --SoundMode_standart
899         rcButton = mvk_left &
900         UISoundMenu = SoundMenu_soundMode &
901         UImenuStates = UserAudioMenuUI &
902         UISoundMode = SoundMode_standart &
903         isClicked = TRUE : SoundMode_user;
904
905         rcButton = mvk_right &
906         UISoundMenu = SoundMenu_soundMode &
907         UImenuStates = UserAudioMenuUI &
908         UISoundMode = SoundMode_standart &
909

```

```

910         isClicked = TRUE : SoundMode_movie;
911
912     --SoundMode_movie
913         rcButton = mvk_left &
914         UISoundMenu = SoundMenu_soundMode &
915         UImenuStates = UserAudioMenuUI &
916         UISoundMode = SoundMode_movie &
917         isClicked = TRUE : SoundMode_standart;
918
919         rcButton = mvk_right &
920         UISoundMenu = SoundMenu_soundMode &
921         UImenuStates = UserAudioMenuUI &
922         UISoundMode = SoundMode_movie &
923         isClicked = TRUE : SoundMode_natural;
924
925     --SoundMode_natural
926         rcButton = mvk_left &
927         UISoundMenu = SoundMenu_soundMode &
928         UImenuStates = UserAudioMenuUI &
929         UISoundMode = SoundMode_natural &
930         isClicked = TRUE : SoundMode_movie;
931
932         rcButton = mvk_right &
933         UISoundMenu = SoundMenu_soundMode &
934         UImenuStates = UserAudioMenuUI &
935         UISoundMode = SoundMode_natural &
936         isClicked = TRUE : SoundMode_speech;
937
938     --SoundMode_speech
939         rcButton = mvk_left &
940         UISoundMenu = SoundMenu_soundMode &
941         UImenuStates = UserAudioMenuUI &
942         UISoundMode = SoundMode_speech &
943         isClicked = TRUE : SoundMode_natural;
944
945         rcButton = mvk_right &
946         UISoundMenu = SoundMenu_soundMode &
947         UImenuStates = UserAudioMenuUI &
948         UISoundMode = SoundMode_speech &
949         isClicked = TRUE : SoundMode_music;
950
951     --SoundMode_music
952         rcButton = mvk_left &
953         UISoundMenu = SoundMenu_soundMode &
954         UImenuStates = UserAudioMenuUI &
955         UISoundMode = SoundMode_music &
956         isClicked = TRUE : SoundMode_speech;
957
958         rcButton = mvk_right &
959         UISoundMenu = SoundMenu_soundMode &
960         UImenuStates = UserAudioMenuUI &
961         UISoundMode = SoundMode_music &
962         isClicked = TRUE : SoundMode_user;
963
964         TRUE:UISoundMode;
965     esac;
966
967     next (isClicked) :=
968     case
969         --(rcButton = mvk_left | mvk_right)&
970         (UISoundMenu = SoundMenu_soundMode)&

```

```

971         (UImenuStates = UserAudioMenuUI) &
972         (
973             (UISoundMode = SoundMode_user) |
974             (UISoundMode = SoundMode_standart) |
975             (UISoundMode = SoundMode_movie) |
976             (UISoundMode = SoundMode_natural) |
977             (UISoundMode = SoundMode_speech) |
978             (UISoundMode = SoundMode_music)
979         ) &
980         (isClicked = TRUE):FALSE;
981     TRUE:isClicked;
982 esac;
983 FAIRNESS
984     running
985
986
987 --Navigation between sound menu items by rcButton left and right
988 MODULE SoundMenu(isClicked,rcButton,UImenuStates,UISoundMenu)
989 ASSIGN
990
991 --back ve manu tusşları ile menudan çıkmayı da belirt
992     next (UISoundMenu):=
993     case
994         rcButton = mvk_down &
995         UISoundMenu = SoundMenu_none &
996         UImenuStates = UserAudioMenuUI &
997         isClicked = TRUE : SoundMenu_soundMode;
998
999 --SoundMenu_soundMode*****
1000     rcButton = mvk_down &
1001     UISoundMenu = SoundMenu_soundMode &
1002     UImenuStates = UserAudioMenuUI &
1003     isClicked = TRUE : SoundMenu_nightMode;
1004
1005     rcButton = mvk_up &
1006     UISoundMenu = SoundMenu_soundMode &
1007     UImenuStates = UserAudioMenuUI &
1008     isClicked = TRUE : SoundMenu_RestoretoDefault;
1009
1010 --SoundMenu_nightMode*****
1011     rcButton = mvk_down &
1012     UISoundMenu = SoundMenu_nightMode &
1013     UImenuStates = UserAudioMenuUI &
1014     isClicked = TRUE : SoundMenu_bluetooth;
1015
1016     rcButton = mvk_up &
1017     UISoundMenu = SoundMenu_nightMode &
1018     UImenuStates = UserAudioMenuUI &
1019     isClicked = TRUE : SoundMenu_soundMode;
1020
1021 --SoundMenu_bluetooth*****
1022     rcButton = mvk_down &
1023     UISoundMenu = SoundMenu_bluetooth &
1024     UImenuStates = UserAudioMenuUI &
1025     isClicked = TRUE : SoundMenu_TVLocation;
1026
1027     rcButton = mvk_up &
1028     UISoundMenu = SoundMenu_bluetooth &
1029     UImenuStates = UserAudioMenuUI &
1030     isClicked = TRUE : SoundMenu_nightMode;
1031

```

```

1032
1033 --SoundMenu_TVLocation*****
1034
1035         rcButton = mvk_down &
1036         UISoundMenu = SoundMenu_TVLocation &
1037         UImenuStates = UserAudioMenuUI &
1038         isClicked = TRUE : SoundMenu_soundType;
1039
1040         rcButton = mvk_up &
1041         UISoundMenu = SoundMenu_TVLocation &
1042         UImenuStates = UserAudioMenuUI &
1043         isClicked = TRUE : SoundMenu_bluetooth;
1044 --SoundMenu_soundType*****
1045
1046         rcButton = mvk_down &
1047         UISoundMenu = SoundMenu_soundType &
1048         UImenuStates = UserAudioMenuUI &
1049         isClicked = TRUE : SoundMenu_graphEq;
1050
1051         rcButton = mvk_up &
1052         UISoundMenu = SoundMenu_soundType &
1053         UImenuStates = UserAudioMenuUI &
1054         isClicked = TRUE : SoundMenu_TVLocation;
1055 --SoundMenu_graphEq*****
1056         rcButton = mvk_down &
1057         UISoundMenu = SoundMenu_graphEq &
1058         UImenuStates = UserAudioMenuUI &
1059         isClicked = TRUE : SoundMenu_avl;
1060
1061         rcButton = mvk_up &
1062         UISoundMenu = SoundMenu_graphEq &
1063         UImenuStates = UserAudioMenuUI &
1064         isClicked = TRUE : SoundMenu_soundType;
1065
1066 --SoundMenu_avl*****
1067         rcButton = mvk_down &
1068         UISoundMenu = SoundMenu_avl &
1069         UImenuStates = UserAudioMenuUI &
1070         isClicked = TRUE : SoundMenu_HPvolume;
1071
1072         rcButton = mvk_up &
1073         UISoundMenu = SoundMenu_avl &
1074         UImenuStates = UserAudioMenuUI &
1075         isClicked = TRUE : SoundMenu_graphEq;
1076
1077 --SoundMenu_HPvolume*****
1078         rcButton = mvk_down &
1079         UISoundMenu = SoundMenu_HPvolume &
1080         UImenuStates = UserAudioMenuUI &
1081         isClicked = TRUE : SoundMenu_TVSpeaker;
1082
1083         rcButton = mvk_up &
1084         UISoundMenu = SoundMenu_HPvolume &
1085         UImenuStates = UserAudioMenuUI &
1086         isClicked = TRUE : SoundMenu_avl;
1087
1088 --SoundMenu_TVSpeaker*****
1089         rcButton = mvk_down &
1090         UISoundMenu = SoundMenu_TVSpeaker &
1091         UImenuStates = UserAudioMenuUI &
1092         isClicked = TRUE : SoundMenu_ARC;

```

```

1093
1094         rcButton = mvk_up &
1095         UISoundMenu = SoundMenu_TVSpeaker &
1096         UImenuStates = UserAudioMenuUI &
1097         isClicked = TRUE : SoundMenu_HPvolume;
1098
1099 --SoundMenu_ARC*****
1100         rcButton = mvk_down &
1101         UISoundMenu = SoundMenu_ARC &
1102         UImenuStates = UserAudioMenuUI &
1103         isClicked = TRUE : SoundMenu_ScartVolume;
1104
1105         rcButton = mvk_up &
1106         UISoundMenu = SoundMenu_ARC &
1107         UImenuStates = UserAudioMenuUI &
1108         isClicked = TRUE : SoundMenu_TVSpeaker;
1109
1110 --SoundMenu_ScartVolume*****
1111         rcButton = mvk_down &
1112         UISoundMenu = SoundMenu_ScartVolume &
1113         UImenuStates = UserAudioMenuUI &
1114         isClicked = TRUE : SoundMenu_OpticOut;
1115
1116         rcButton = mvk_up &
1117         UISoundMenu = SoundMenu_ScartVolume &
1118         UImenuStates = UserAudioMenuUI &
1119         isClicked = TRUE : SoundMenu_ARC;
1120
1121 --SoundMenu_OpticOut*****
1122         rcButton = mvk_down &
1123         UISoundMenu = SoundMenu_OpticOut &
1124         UImenuStates = UserAudioMenuUI &
1125         isClicked = TRUE : SoundMenu_RestoretoDefault;
1126
1127         rcButton = mvk_up &
1128         UISoundMenu = SoundMenu_OpticOut &
1129         UImenuStates = UserAudioMenuUI &
1130         isClicked = TRUE : SoundMenu_ScartVolume;
1131
1132 --SoundMenu_RestoretoDefault*****
1133
1134         rcButton = mvk_down &
1135         UISoundMenu = SoundMenu_RestoretoDefault &
1136         UImenuStates = UserAudioMenuUI &
1137         isClicked = TRUE : SoundMenu_soundMode;
1138
1139         rcButton = mvk_up &
1140         UISoundMenu = SoundMenu_RestoretoDefault &
1141         UImenuStates = UserAudioMenuUI &
1142         isClicked = TRUE : SoundMenu_OpticOut;
1143 --else-----
1144         TRUE:UISoundMenu;
1145
1146     esac;
1147
1148     --basılabilmesi için işlem bitti flagi de koy
1149
1150     next (isClicked):=
1151     case
1152         --(rcButton = mvk_down | mvk_up) &
1153         (UISoundMenu = SoundMenu_none|

```

```

1154             UISoundMenu = SoundMenu_soundMode|
1155             UISoundMenu = SoundMenu_nightMode|
1156             UISoundMenu = SoundMenu_bluetooth|
1157             UISoundMenu = SoundMenu_TVLocation|
1158             UISoundMenu = SoundMenu_soundType|
1159             UISoundMenu = SoundMenu_graphEq|
1160             UISoundMenu = SoundMenu_avl|
1161             UISoundMenu = SoundMenu_HPvolume|
1162             UISoundMenu = SoundMenu_TVSpeaker|
1163             UISoundMenu = SoundMenu_ARC|
1164             UISoundMenu = SoundMenu_ScartVolume|
1165             UISoundMenu = SoundMenu_OpticOut|
1166             UISoundMenu = SoundMenu_RestoretoDefault
1167         ) &
1168         (UImenuStates = UserAudioMenuUI) &
1169         (isClicked = TRUE):FALSE;
1170     TRUE:isClicked;
1171     esac;
1172
1173 FAIRNESS
1174     running
1175
1176
1177 --This module is used to change the DVB-T TV Channels
1178 MODULE
1179     changeDVB_T_Program(TV_InputSource,UImenuStates,isClicked,rcButton,d
1180     vb_t_channel)
1181
1182 ASSIGN
1183     next(dvb_t_channel) :=
1184         case
1185             TV_InputSource = InputSource_TERRESTRIAL &
1186             isClicked=TRUE & rcButton = mvk_channelPlus & UImenuStates = none &
1187             dvb_t_channel = dvbt_ch_one :dvbt_ch_two;
1188             TV_InputSource = InputSource_TERRESTRIAL &
1189             isClicked=TRUE & rcButton = mvk_channelPlus & UImenuStates = none &
1190             dvb_t_channel = dvbt_ch_two :dvbt_ch_three;
1191             TV_InputSource = InputSource_TERRESTRIAL &
1192             isClicked=TRUE & rcButton = mvk_channelPlus & UImenuStates = none &
1193             dvb_t_channel = dvbt_ch_three :dvbt_ch_one;
1194             TV_InputSource = InputSource_TERRESTRIAL &
1195             isClicked=TRUE & rcButton = mvk_channelMinus & UImenuStates = none &
1196             dvb_t_channel = dvbt_ch_one :dvbt_ch_three;
1197             TV_InputSource = InputSource_TERRESTRIAL &
1198             isClicked=TRUE & rcButton = mvk_channelMinus & UImenuStates = none &
1199             dvb_t_channel = dvbt_ch_two :dvbt_ch_one;
1200             TV_InputSource = InputSource_TERRESTRIAL &
1201             isClicked=TRUE & rcButton = mvk_channelMinus & UImenuStates = none &
1202             dvb_t_channel = dvbt_ch_three :dvbt_ch_two;
1203             TRUE:dvb_t_channel;
1204         esac;
1205
1206
1207
1208 FAIRNESS
1209     running
1210
1211
1212 --This module is used to change the DVB-C TV Channels

```

```

1213 MODULE
1214 changeDVB_C_Program(TV_InputSource,UImenuStates,isClicked,rcButton,d
1215 vb_c_channel)
1216
1217 ASSIGN
1218     next(dvb_c_channel) :=
1219         case
1220             TV_InputSource = InputSource_CABLE &
1221             isClicked=TRUE & rcButton = mvk_channelPlus & UImenuStates = none &
1222             dvb_c_channel = dvbc_ch_one :dvbc_ch_two;
1223             TV_InputSource = InputSource_CABLE &
1224             isClicked=TRUE & rcButton = mvk_channelPlus & UImenuStates = none &
1225             dvb_c_channel = dvbc_ch_two :dvbc_ch_three;
1226             TV_InputSource = InputSource_CABLE &
1227             isClicked=TRUE & rcButton = mvk_channelPlus & UImenuStates = none &
1228             dvb_c_channel = dvbc_ch_three :dvbc_ch_one;
1229             TV_InputSource = InputSource_CABLE &
1230             isClicked=TRUE & rcButton = mvk_channelMinus & UImenuStates = none &
1231             dvb_c_channel = dvbc_ch_one :dvbc_ch_three;
1232             TV_InputSource = InputSource_CABLE &
1233             isClicked=TRUE & rcButton = mvk_channelMinus & UImenuStates = none &
1234             dvb_c_channel = dvbc_ch_two :dvbc_ch_one;
1235             TV_InputSource = InputSource_CABLE &
1236             isClicked=TRUE & rcButton = mvk_channelMinus & UImenuStates = none &
1237             dvb_c_channel = dvbc_ch_three :dvbc_ch_two;
1238             TRUE:dvb_c_channel;
1239         esac;
1240
1241
1242
1243 FAIRNESS
1244     running
1245
1246 --This module is used to change the DVB-S TV Channels
1247 MODULE
1248 changeDVB_S_Program(TV_InputSource,UImenuStates,isClicked,rcButton,d
1249 vb_s_channel)
1250
1251 ASSIGN
1252     next(dvb_s_channel) :=
1253         case
1254             TV_InputSource = InputSource_SATELLITE &
1255             isClicked=TRUE & rcButton = mvk_channelPlus & UImenuStates = none &
1256             dvb_s_channel = dvbs_ch_one :dvbs_ch_two;
1257             TV_InputSource = InputSource_SATELLITE &
1258             isClicked=TRUE & rcButton = mvk_channelPlus & UImenuStates = none &
1259             dvb_s_channel = dvbs_ch_two :dvbs_ch_three;
1260             TV_InputSource = InputSource_SATELLITE &
1261             isClicked=TRUE & rcButton = mvk_channelPlus & UImenuStates = none &
1262             dvb_s_channel = dvbs_ch_three :dvbs_ch_one;
1263             TV_InputSource = InputSource_SATELLITE &
1264             isClicked=TRUE & rcButton = mvk_channelMinus & UImenuStates = none &
1265             dvb_s_channel = dvbs_ch_one :dvbs_ch_three;
1266             TV_InputSource = InputSource_SATELLITE &
1267             isClicked=TRUE & rcButton = mvk_channelMinus & UImenuStates = none &
1268             dvb_s_channel = dvbs_ch_two :dvbs_ch_one;
1269             TV_InputSource = InputSource_SATELLITE &
1270             isClicked=TRUE & rcButton = mvk_channelMinus & UImenuStates = none &
1271             dvb_s_channel = dvbs_ch_three :dvbs_ch_two;
1272             TRUE:dvb_s_channel;
1273         esac;

```

```

1274
1275
1276
1277
1278
1279
1280 FAIRNESS
1281     running
1282
1283
1284 --This module is our user-agent module. This is non-deterministic
1285 user-agent
1286 MODULE userClick(isClicked,rcButton,userClickOrder)
1287     ASSIGN
1288         next(rcButton) :=
1289             case
1290                 isClicked = FALSE : {mvk_power,          --1
1291
1292 mvk_number, --2
1293 mvk_FAV,
1294 --3
1295 mvk_3D,
1296 --4
1297 mvk_RED,
1298 --5
1299 mvk_GREEN, --6
1300 mvk_YELLOW, --7
1301 mvk_BLUE,
1302 --8
1303 mvk_info,
1304 --9
1305 mvk_exit,
1306 --a
1307 mvk_menu,
1308 --b
1309 mvk_back,
1310 --c
1311 mvk_tools, --d
1312 mvk_guide, --e
1313 mvk_volumePlus, --f
1314 mvk_volumeMinus, --10
1315 mvk_channelPlus, --11
1316 mvk_channelMinus, --12
1317 mvk_sourceChange, --13
1318 mvk_mute,
1319 --14
1320 mvk_play,
1321 --15
1322 mvk_stop,
1323 --16
1324 mvk_pause, --17

```

```

1335
1336     mvk_backward,          --18
1337
1338     mvk_forward,           --19
1339
1340     mvk_previous,          --1a
1341                                     mvk_next,
1342                                     --1b
1343
1344     mvk_audio,              --1c
1345
1346     mvk_subtitle,          --1d
1347                                     mvk_pvr
1348     ,                       --1e
1349                                     mvk_up,
1350                                     --1f
1351                                     mvk_down,
1352                                     --20
1353                                     mvk_left,
1354                                     --21
1355
1356     mvk_right,              --22
1357                                     mvk_ok
1358 };                           --23
1359     TRUE: rcButton;
1360     esac;
1361
1362     next (isClicked):=
1363         case
1364             isClicked = FALSE: TRUE;
1365             TRUE:isClicked;
1366         esac;
1367
1368 FAIRNESS
1369     running
1370
1371 -- Set isClicked to False for the rcButtons which are not defined
1372 yet.
1373 MODULE
1374 eliminateUnnecessaryUserClicks(isClicked,rcButton,UImenuStates)--bak
1375 buraaa
1376 ASSIGN
1377     next(isClicked):=
1378         case
1379             UImenuStates = none & (rcButton!=mvk_menu |
1380 rcButton!=mvk_sourceChange | rcButton!=mvk_channelPlus |
1381 rcButton!=mvk_channelMinus | rcButton!=mvk_mute |
1382 rcButton!=mvk_volumePlus | rcButton!=mvk_volumeMinus) : FALSE;
1383             UImenuStates = SourceMenuUI & (rcButton!=mvk_left
1384 | rcButton!=mvk_right | rcButton!=mvk_sourceChange |
1385 rcButton!=mvk_ok | rcButton!=mvk_back): FALSE;
1386             UImenuStates = UserAudioMenuUI &
1387 (rcButton!=mvk_left | rcButton!=mvk_right | rcButton!=mvk_menu|
1388 rcButton!=mvk_ok | rcButton!=mvk_back): FALSE;
1389
1390             TRUE:isClicked;
1391         esac;
1392
1393 FAIRNESS
1394     running
1395

```

```

1396
1397 --Enable/Disable the source menu.
1398 MODULE
1399 source_click(isClicked,rcButton,UImenuStates,SourceMenuItems,IstTVAud
1400 ioInitied)
1401     ASSIGN
1402         next (SourceMenuItems):=
1403         case
1404
1405
1406             (isClicked = TRUE) &
1407             (UImenuStates = none)&
1408             IstTVAudioInitied = TRUE &
1409             (rcButton = mvk_sourceChange):none;
1410
1411             (isClicked = TRUE) &
1412             (UImenuStates = SourceMenuUI)&
1413             IstTVAudioInitied = TRUE &
1414             (rcButton = mvk_sourceChange):none;
1415
1416             (isClicked = TRUE) &
1417             (UImenuStates = SourceMenuUI)&
1418             IstTVAudioInitied = TRUE &
1419             (rcButton = mvk_back):none;
1420
1421
1422
1423             TRUE:SourceMenuItems;
1424         esac;
1425
1426
1427
1428     next (UImenuStates):=
1429     case
1430         (isClicked = TRUE) &
1431         (UImenuStates = none)&
1432         IstTVAudioInitied = TRUE &
1433         (rcButton = mvk_sourceChange):SourceMenuUI;
1434
1435         (isClicked = TRUE) &
1436         (UImenuStates = SourceMenuUI)&
1437         IstTVAudioInitied = TRUE &
1438         (rcButton = mvk_sourceChange):none;
1439
1440         (isClicked = TRUE) &
1441         (UImenuStates = SourceMenuUI)&
1442         IstTVAudioInitied = TRUE &
1443         (rcButton = mvk_back):none;
1444         TRUE:UImenuStates;
1445     esac;
1446
1447     next (isClicked):=
1448     case
1449         IstTVAudioInitied = TRUE &(isClicked = TRUE) &(
1450         ((UImenuStates = none)&(rcButton =
1451 mvk_sourceChange)) |
1452         ((UImenuStates = SourceMenuUI)& ((rcButton =
1453 mvk_sourceChange)|(rcButton = mvk_back)))) ) :FALSE;
1454         TRUE: isClicked;
1455
1456     esac;

```

```

1457
1458
1459
1460
1461 FAIRNESS
1462     running
1463
1464
1465
1466
1467
1468
1469 -- navigation on the source menu items
1470 MODULE
1471     source_change(isClicked,rcButton,UImenuStates,SourceMenuItems)
1472
1473 ASSIGN
1474     next(SourceMenuItems) :=
1475
1476         case
1477             rcButton = mvk_right &
1478             SourceMenuItems = none &
1479             UImenuStates = SourceMenuUI &
1480             isClicked = TRUE : SourceMenu_HDMI;
1481
1482         -----SourceMenu_HDMI
1483
1484             rcButton = mvk_right &
1485             SourceMenuItems = SourceMenu_HDMI &
1486             UImenuStates = SourceMenuUI &
1487             isClicked = TRUE : SourceMenu_SCART;
1488
1489             rcButton = mvk_left &
1490             SourceMenuItems = SourceMenu_HDMI &
1491             UImenuStates = SourceMenuUI &
1492             isClicked = TRUE : SourceMenu_PC;
1493
1494         -----SourceMenu_SCART
1495
1496             rcButton = mvk_right &
1497             SourceMenuItems = SourceMenu_SCART &
1498             UImenuStates = SourceMenuUI &
1499             isClicked = TRUE : SourceMenu_COMPONENT;
1500
1501             rcButton = mvk_left &
1502             SourceMenuItems = SourceMenu_SCART &
1503             UImenuStates = SourceMenuUI &
1504             isClicked = TRUE : SourceMenu_HDMI;
1505
1506         -----SourceMenu_COMPONENT
1507
1508             rcButton = mvk_right &
1509             SourceMenuItems = SourceMenu_COMPONENT &
1510             UImenuStates = SourceMenuUI &
1511             isClicked = TRUE : SourceMenu_CABLE;
1512
1513             rcButton = mvk_left &
1514             SourceMenuItems = SourceMenu_COMPONENT &
1515             UImenuStates = SourceMenuUI &
1516             isClicked = TRUE : SourceMenu_SCART;
1517

```

```

1518 -----SourceMenu_CABLE
1519
1520 rcButton = mvk_right &
1521 SourceMenuItems = SourceMenu_CABLE &
1522 UImenuStates = SourceMenuUI &
1523 isClicked = TRUE : SourceMenu_TERRESTRIAL;
1524
1525 rcButton = mvk_left &
1526 SourceMenuItems = SourceMenu_CABLE &
1527 UImenuStates = SourceMenuUI &
1528 isClicked = TRUE : SourceMenu_COMPONENT;
1529
1530 -----SourceMenu_TERRESTRIAL
1531
1532 rcButton = mvk_right &
1533 SourceMenuItems = SourceMenu_TERRESTRIAL &
1534 UImenuStates = SourceMenuUI &
1535 isClicked = TRUE : SourceMenu_SATELLITE;
1536
1537 rcButton = mvk_left &
1538 SourceMenuItems = SourceMenu_TERRESTRIAL &
1539 UImenuStates = SourceMenuUI &
1540 isClicked = TRUE : SourceMenu_CABLE;
1541
1542 -----SourceMenu_SATELLITE
1543
1544 rcButton = mvk_right &
1545 SourceMenuItems = SourceMenu_SATELLITE &
1546 UImenuStates = SourceMenuUI &
1547 isClicked = TRUE : SourceMenu_ATV;
1548
1549 rcButton = mvk_left &
1550 SourceMenuItems = SourceMenu_SATELLITE &
1551 UImenuStates = SourceMenuUI &
1552 isClicked = TRUE : SourceMenu_TERRESTRIAL;
1553
1554 -----SourceMenu_ATV
1555
1556 rcButton = mvk_right &
1557 SourceMenuItems = SourceMenu_ATV &
1558 UImenuStates = SourceMenuUI &
1559 isClicked = TRUE : SourceMenu_USB;
1560
1561 rcButton = mvk_left &
1562 SourceMenuItems = SourceMenu_ATV &
1563 UImenuStates = SourceMenuUI &
1564 isClicked = TRUE : SourceMenu_SATELLITE;
1565
1566 -----SourceMenu_USB
1567
1568 rcButton = mvk_right &
1569 SourceMenuItems = SourceMenu_USB &
1570 UImenuStates = SourceMenuUI &
1571 isClicked = TRUE : SourceMenu_PC;
1572
1573 rcButton = mvk_left &
1574 SourceMenuItems = SourceMenu_USB &
1575 UImenuStates = SourceMenuUI &
1576 isClicked = TRUE : SourceMenu_ATV;
1577
1578

```

```

1579             -----SourceMenu_PC
1580
1581             rcButton = mvk_right &
1582             SourceMenuItems = SourceMenu_PC &
1583             UImenuStates = SourceMenuUI &
1584             isClicked = TRUE : SourceMenu_HDMI;
1585
1586             TRUE : SourceMenuItems;
1587         esac;
1588
1589     next (isClicked):=
1590         case
1591             (rcButton = mvk_right|rcButton = mvk_left) &
1592             SourceMenuItems !=none &
1593             UImenuStates = SourceMenuUI &
1594             isClicked = TRUE:FALSE;
1595             TRUE:isClicked;
1596         esac;
1597
1598 FAIRNESS
1599     running
1600
1601
1602
1603 -- do the source change job
1604 MODULE
1605 source_change_monitor(TV_InputSource,isClicked,rcButton,UImenuStates
1606 ,SourceMenuItems,IstVAudioInitied,dvb_t_channel,dvb_c_channel,dvb_s_
1607 channel)
1608     ASSIGN
1609
1610     next (TV_InputSource):=
1611         case
1612             IstVAudioInitied = TRUE &
1613             isClicked =TRUE & rcButton = mvk_ok &
1614             UImenuStates = SourceMenuUI &
1615 SourceMenuItems =SourceMenu_PC :InputSource_PC ;
1616
1617             IstVAudioInitied = TRUE &
1618             isClicked =TRUE & rcButton = mvk_ok &
1619             UImenuStates = SourceMenuUI &
1620 SourceMenuItems =SourceMenu_HDMI :InputSource_HDMI ;
1621
1622             IstVAudioInitied = TRUE &
1623             isClicked =TRUE & rcButton = mvk_ok &
1624             UImenuStates = SourceMenuUI &
1625 SourceMenuItems =SourceMenu_ATV :InputSource_ATV ;
1626
1627             IstVAudioInitied = TRUE &
1628             isClicked =TRUE & rcButton = mvk_ok &
1629             UImenuStates = SourceMenuUI &
1630 SourceMenuItems =SourceMenu_SATELLITE :InputSource_SATELLITE ;
1631
1632             IstVAudioInitied = TRUE &
1633             isClicked =TRUE & rcButton = mvk_ok &
1634             UImenuStates = SourceMenuUI &
1635 SourceMenuItems =SourceMenu_TERRESTRIAL :InputSource_TERRESTRIAL ;
1636
1637             IstVAudioInitied = TRUE &
1638             isClicked =TRUE & rcButton = mvk_ok &

```

```

1639             UImenuStates = SourceMenuUI &
1640 SourceMenuItems =SourceMenu_CABLE :InputSource_CABLE ;
1641
1642             IstVAudioInitied = TRUE &
1643             isClicked =TRUE & rcButton = mvk_ok &
1644             UImenuStates = SourceMenuUI &
1645 SourceMenuItems =SourceMenu_COMPONENT :InputSource_COMPONENT ;
1646
1647             IstVAudioInitied = TRUE &
1648             isClicked =TRUE & rcButton = mvk_ok &
1649             UImenuStates = SourceMenuUI &
1650 SourceMenuItems =SourceMenu_SCART :InputSource_SCART ;
1651
1652
1653             IstVAudioInitied = TRUE &
1654             isClicked =TRUE & rcButton = mvk_ok &
1655             UImenuStates = SourceMenuUI &
1656 SourceMenuItems =SourceMenu_USB :InputSource_USB ;
1657
1658             TRUE:TV_InputSource;
1659         esac;
1660     next (dvb_s_channel) :=
1661     case
1662         IstVAudioInitied = TRUE &
1663         isClicked =TRUE & rcButton = mvk_ok &
1664         UImenuStates = SourceMenuUI &
1665 SourceMenuItems =SourceMenu_SATELLITE :dvbs_ch_one;
1666         TRUE:dvb_s_channel;
1667     esac;
1668     next (dvb_t_channel) :=
1669     case
1670         IstVAudioInitied = TRUE &
1671         isClicked =TRUE & rcButton = mvk_ok &
1672         UImenuStates = SourceMenuUI &
1673 SourceMenuItems =SourceMenu_TERRESTRIAL :dvbt_ch_one;
1674         TRUE:dvb_t_channel;
1675     esac;
1676     next (dvb_c_channel) :=
1677     case
1678         IstVAudioInitied = TRUE &
1679         isClicked =TRUE & rcButton = mvk_ok &
1680         UImenuStates = SourceMenuUI &
1681 SourceMenuItems =SourceMenu_CABLE :dvbc_ch_one;
1682         TRUE:dvb_c_channel;
1683     esac;
1684
1685
1686     next (isClicked) :=
1687     case
1688         IstVAudioInitied = TRUE &
1689         isClicked =TRUE & rcButton = mvk_ok &
1690         UImenuStates = SourceMenuUI &
1691 SourceMenuItems !=none : FALSE;
1692         TRUE:isClicked;
1693     esac;
1694
1695 FAIRNESS
1696     running
1697
1698 -- Adjust the TV volume level according to the user clicks.

```

```

1699 MODULE
1700 Volume(isClicked,rcButton,UImenuStates,SoundExist,isTVMute,TVVolumeL
1701 evel,IstTVAudioInitied)
1702
1703 ASSIGN
1704
1705     next (TVVolumeLevel):=
1706     case
1707         (rcButton = mvk_volumePlus) &
1708         UImenuStates = none &
1709         IstTVAudioInitied = TRUE &
1710         TVVolumeLevel != 21 &
1711         isClicked = TRUE : TVVolumeLevel+1;
1712
1713         (rcButton = mvk_volumeMinus) &
1714         UImenuStates = none &
1715         IstTVAudioInitied = TRUE &
1716         TVVolumeLevel != 0 &
1717         isClicked = TRUE : TVVolumeLevel+(-1);
1718
1719         TRUE : TVVolumeLevel;
1720     esac;
1721
1722     next (isTVMute):=
1723     case
1724         (rcButton = mvk_mute) &
1725         UImenuStates = none &
1726         IstTVAudioInitied = TRUE &
1727         isTVMute = TRUE &
1728         isClicked = TRUE : FALSE;
1729
1730         (rcButton = mvk_mute) &
1731         UImenuStates = none &
1732         IstTVAudioInitied = TRUE &
1733         isTVMute = FALSE &
1734         isClicked = TRUE : TRUE;
1735
1736         TRUE : isTVMute;
1737     esac;
1738
1739     -- If TV is muted, then there must be no sound
1740     next (SoundExist):=
1741     case
1742         (rcButton = mvk_mute) &
1743         UImenuStates = none &
1744         IstTVAudioInitied = TRUE &
1745         isTVMute = TRUE &
1746         isClicked = TRUE : FALSE;
1747
1748         (rcButton = mvk_mute) &
1749         UImenuStates = none &
1750         IstTVAudioInitied = TRUE &
1751         isTVMute = FALSE &
1752         isClicked = TRUE : TRUE;
1753
1754         (rcButton = mvk_volumeMinus | rcButton = mvk_volumePlus)
1755     &
1756         UImenuStates = none &
1757         IstTVAudioInitied = TRUE &
1758         TVVolumeLevel != 0 &
1759         isClicked = TRUE : TRUE;

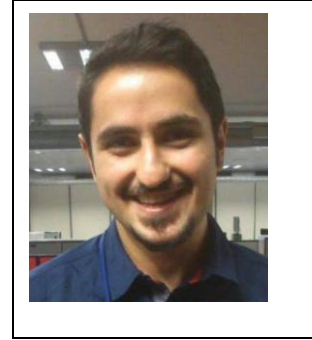
```

```

1760
1761             (rcButton = mvk_volumeMinus | rcButton = mvk_volumePlus)
1762 &
1763             UImenuStates = none &
1764             IsTVAudioInitied = TRUE &
1765             TVVolumeLevel = 0 &
1766             isClicked = TRUE : FALSE;
1767
1768             TRUE : SoundExist;
1769     esac;
1770
1771     next (isClicked):=
1772     case
1773         (rcButton = mvk_volumeMinus | rcButton = mvk_volumePlus
1774 | rcButton =mvk_mute) &
1775         UImenuStates = none &
1776         IsTVAudioInitied = TRUE &
1777         isClicked = TRUE: FALSE;
1778         TRUE: isClicked;
1779     esac;
1780
1781
1782
1783
1784 FAIRNESS
1785     running
1786

```


CURRICULUM VITAE



Name Surname: FURKAN CÖMERT

Place and Date of Birth: ANKARA 03.09.1988

E-Mail: furkan.comert@arcelik.com

EDUCATION: MSc candidate at Computer Engineering
Department of ITU

B.Sc.: Dokuz Eylul University, Electrical and Electronics
Engineering

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS

▪ **Cömert F.**, Ovatman T.(2015). Attacking State Space Explosion Problem In Model Checking Embedded TV Software. *IEEE Transactions on Consumer Electronics*. November 2015

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS :

▪ **Cömert F.**, Erşahin G., Çelebican O., Olçücü H., 2015-10-01 Image display device with automatic sound enhancement function. Patent Publication number: WO2015144243 A1

▪ **Cömert F.**, Ozbilek B., Çelebican O., Erşahin G., Olçücü H., 2015-07-09 Method for operating an image display device with voice-over feature. Patent Publication number: WO2015101395 A1

▪ Erşahin G., Çelebican O., Olçücü H., Altınkök G., **Cömert F.**, 2015-07-02 Image display device with program-based automatic audio signal and subtitle switching function. Patent Publication number: WO2015096871 A1

▪ **Cömert F.**, Çelebican O., Olçücü H.,Altınkök G., Erşahin G.,2015-07-02 Image display device with program-based automatic channel switching function. Patent Publication number:WO 2015096870 A1

▪ Olçücü H.,Altınkök G., Çelebican O., Erşahin G., **Cömert F.**,2014-07-03 A digital broadcast receiving device comprising a signal measurement unit. Patent Publication number:WO2014102379 A1