고장수목을 이용한 테스트 케이스의 안전성 측정

윤상현, 조재연, 유준범
Dependable Software Laboratory
건국대학교

차례

- 서론
- 배경지식
  - 고장수목분석
- 테스트 케이스와 고장수목의 최소 절단 집합의 비교
  - 개요
  - 소프트웨어 요구사항 모델 - 핸드폰 카메라 예제
  - 고장수목분석 최소 절단 집합의 CTL 속성으로의 변환
  - 테스트 케이스에서 SMV 입력 프로그램으로의 변환
  - 테스트 케이스 변환 모델에 대한 모델체킹
- 결론 및 향후 연구

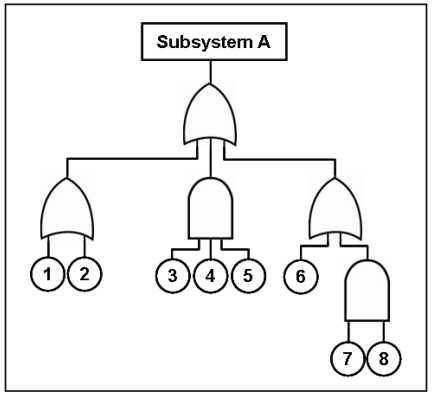
서론

- 테스트 케이스의 품질은 테스팅의 품질 및 효율에 직결된다.
- 테스팅 요구사항에서 고려해야 할 점들은 도메인마다 다르기 때문에 테스트 케이스의 품질을 측정하는 기준을 정하기 어렵다.
- 안전필수 시스템에서 시스템의 품질을 향상시키기 위해 고장수목분석과 같은 안전성 분석 기법이 사용되고 있으며 이를 테스트케이스에서 고려해야 할 품질 속성 중 하나로 생각해볼 수 있다.
- 논문에서의 제안
  - 테스트 케이스가 안전성에 대한 고려를 하였는지 확인하는 기준으로 고장수목분석의 최소 절단 집합을 사용한다.
  - 테스트 케이스와 고장수목의 최소 절단 집합을 각각 SMV 입력 프로그램과 CTL(Computational Tree Logic)으로 변환하여 모델체킹을 한다.

고장수목분석

- 고장수목분석(Fault tree analysis)
  - 시스템을 분석하기 위해 위에서 아래로 연역적으로 고장수목을 그려가며 분석하는 방법
  - Failure가 발생하는 원인이 될 수 있는 모든 이벤트들을 Boolean gate를 이용하여 고장수목을 그린다.
  - 자동화되지 않고 주로 안전성 전문가들이 직접 적용하는 분석방식

- 최소 절단 집합(Minimal cut-set)
  - Failure를 발생시키는 유일한 이벤트들의 집합
  - 복잡한 고장 수목에 대한 분석을 위해 안전성 전문가들이 많이 사용



<subsystem A에 대한 고장수목>

Subsystem A =
(1 | 2) | (3 & 4 & 5) | (6 | (7 & 8))

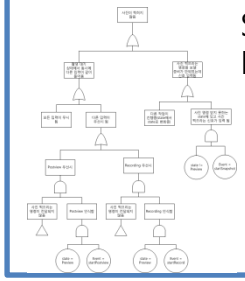
<subsystem A의 최소 절단 집합>

테스트 케이스와 고장수목의 최소 절단 집합의 비교 방법의 개요

사람이 직접 안전성 분석

안전성 분석

Software Fault Tree

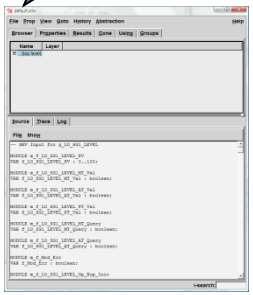


SFT를 참조하여 사람이 생성

CTL property

```
SPEC AG EF 1;
SPEC AG (((state=Preview)&
(event=startSnapshot)&
(event=startRecord)&
(event=stopPostview))
->AX(state=Snapshot));
SPEC AG ((event=startSnapshot)
->AX(state=Snapshot));
```

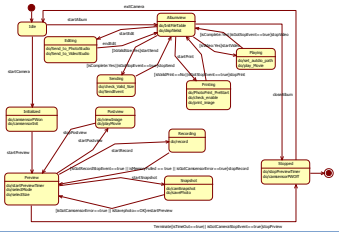
입력



<Cadence SMV>

테스트 케이스를 모델체킹의 대상 모델로, FTA의 결과를 검증속성으로 변환하여 모델체킹

소프트웨어 요구사항 모델

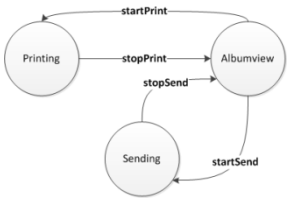


테스트 케이스 자동 생성

테스트 케이스

```
<TestCase id="V 34">
    <StartState value="Albumview"/>
    <Event value="startPrint"/>
    <Action value="do/PhotoPrint_PreStart
do/LoadCurFileInfo"/>
    <NextState value="Printing"/>
    <Event value="postEvent[isComplete]"/>
    <Action value="do/album_InitAppDate
do/Album_PreStart
do/Album_InitFileTable
do/get_file_list
do/disp_file_list"/>
    <EndState value="Albumview"/>
</TestCase>
```

도구를 이용한 자동 변환



오토마타

도구를 이용한 자동 변환

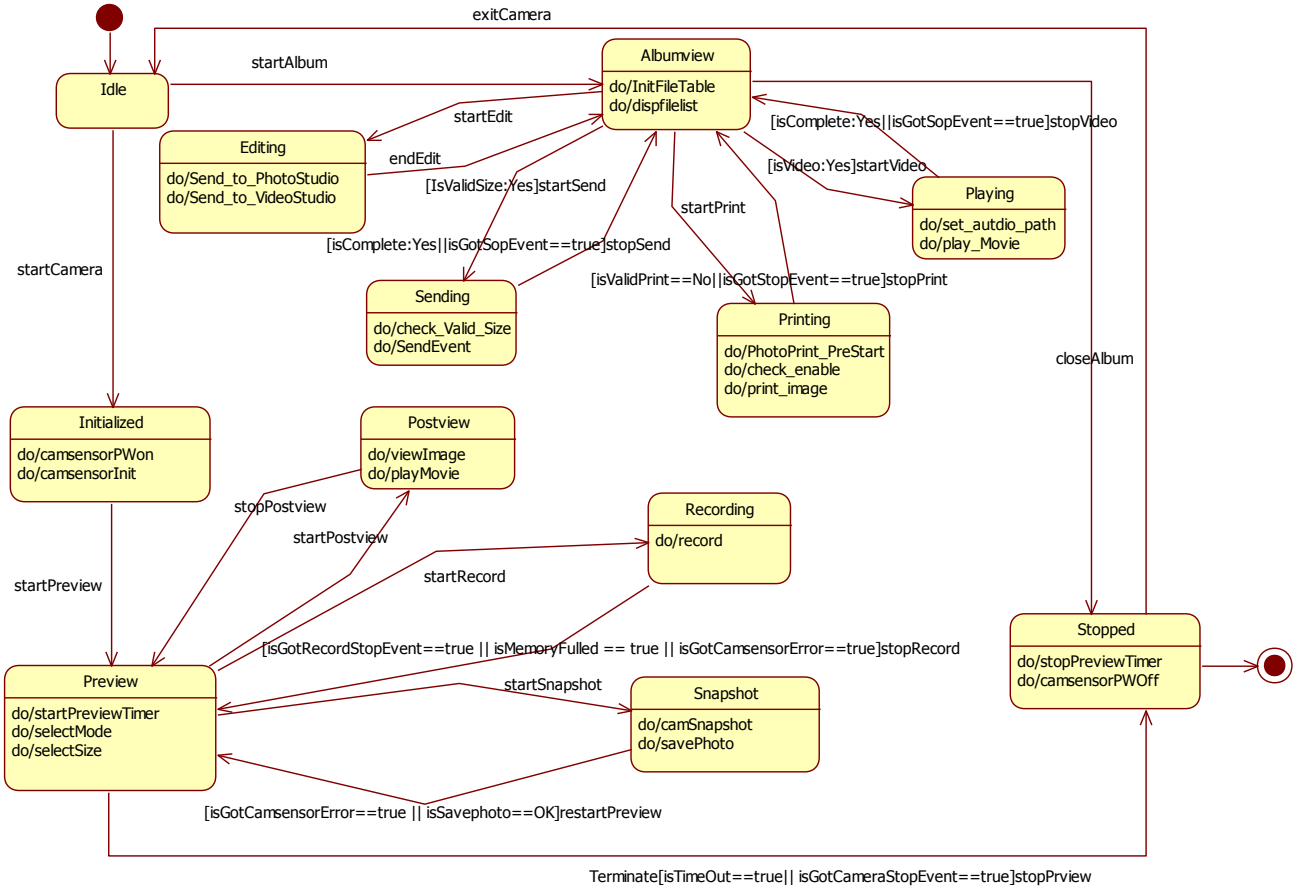
Cadence SMV 입력 언어

입력

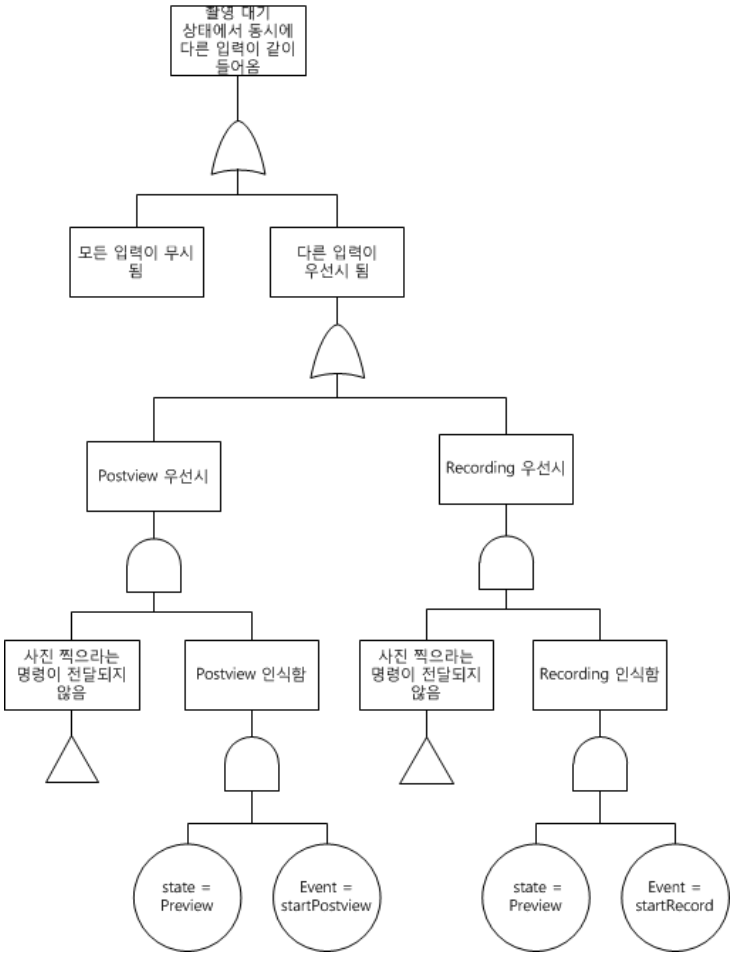
DEPENDABLE SOFTWARE LABORATORY

소프트웨어 요구사항 모델 - 핸드폰 카메라 예제

- State diagram의 형태로 구성되어 있으며 각각의 state와 transition간의 천이에 대한 조건들이 명시되어 있다.

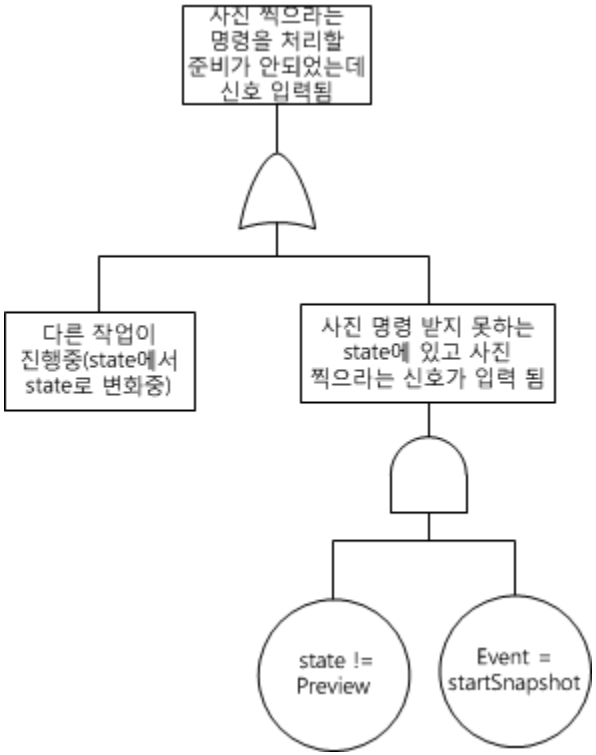


고장수목분석 결과의 CTL 속성으로의 변환(1)



- 동시에 여러 입력이 같이 들어오면 문제가 될 수 있다.
- 여러 입력이 동시에 들어왔을 때 사진을 찍는 state(Sanpshot)로 가는지 확인할 필요가 있다.
- SPEC AG (((state=Preview)&(event=startSnapshot)&(event=startRecord)&(event=stopPostview))->AX(state=Snapshot));

고장수목분석 결과의 CTL 속성으로의 변환(2)



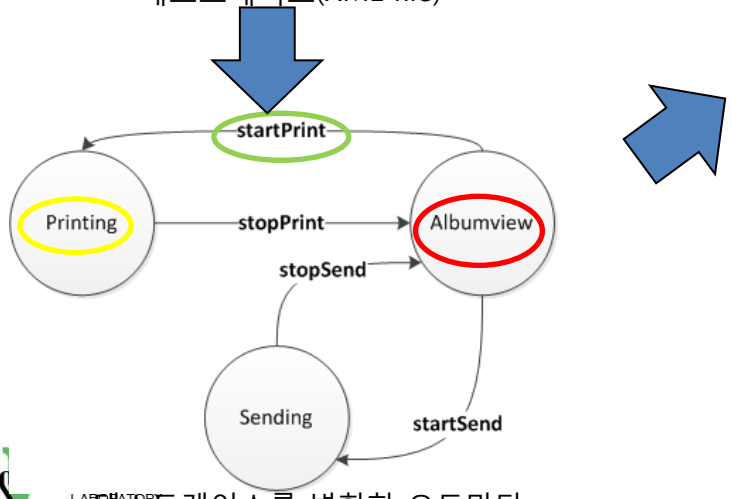
- 사진 촬영 준비 상태(Preview)가 아닌경우에는 사진촬영 입력이 들어와도 처리하지 못할 수 있다.(정상적인 상황)
- 모든 시스템의 상태에서 사진 촬영 이벤트가 발생하면 바로 사진 촬영으로 가는지 확인한다. (만족되면 문제가 되는 상황)
- SPEC AG ((event=startSnapshot) -> AX(state=Snapshot));

테스트 케이스에서 SMV 입력 프로그램으로의 변환

- 테스트 케이스가 소프트웨어 요구사항 모델의 state/transition을 대상으로 하고 있다. 이를 오토마타로 변환하여 다시 SMV 입력 프로그램으로 변환한다.

```
<TestCase id="V_34">
    <StartState value="Albumview"/>
    <Event value="startPrint"/>
    <Action value="do/PhotoPrint_PreStart
do/LoadCurFileInfo"/>
    <NextState value="Printing"/>
    <Event value="postEvent[isComplete]"/>
    <Action value="do/album_InitAppDate
do/Album_PreStart
do/Album_InitFileTable
do/get_file_list
do/disp_file_list"/>
    <EndState value="Albumview"/>
</TestCase>
```

<테스트케이스(XML file)>

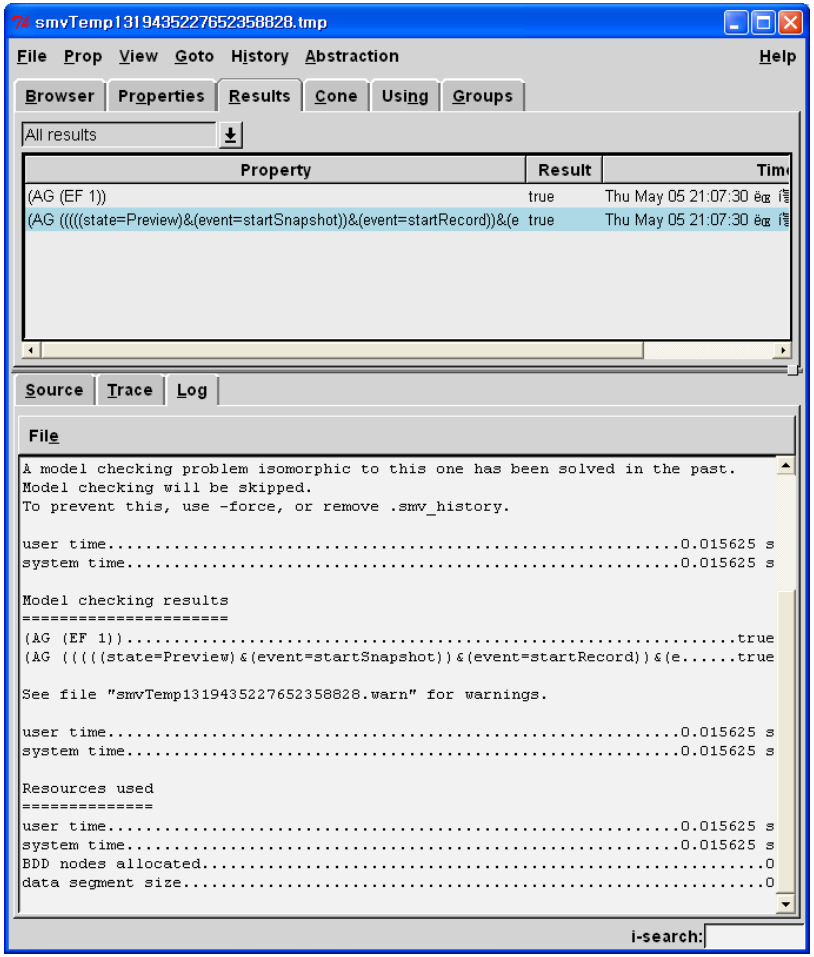


<테스트케이스를 변환한 오토마타>

```
next(state) :=
        switch(state, event){
                (AlbumEditing , stopAlbumEdit):Albumview;
                (Albumview , startAlbumEdit):AlbumEditing;
                (Albumview , startEdit):Editing;
                (Albumview , startVideo):Playing;
                (Albumview , startPrint):Printing;
                (Albumview , send_to_MMS):Sending;
                (Albumview , closeAlbum):Stopped;
                (Editing , stopEdit):Albumview;
                (Idle , startAlbum):Albumview;
                (Idle , startCamera):Initialized;
                (Initialized , oper_preview):Preview;
                (Playing , stopVideo):Albumview;
                (Postview , stopPostview):isComplete?Preview:Postview;
                (Preview , startPostview):Postview;
                (Preview , startRecord):Recording;
                (Preview , startSnapshot):Snapshot;
                (Preview , stopPreview):isTimeOut|isGotCameraStopEvent?Stopped:Preview;
                (Printing , postEvent):isComplete?Albumview:Printing;
                (Recording , restartPreview):isComplete?Preview:Recording;
                (Sending , SendEvent):isComplete?Albumview:Sending;
                (Snapshot , restartpreview):isComplete?Preview:Snapshot;
                (Stopped , exitCamera):Idle;
        };
```

<오토마타를 변환한 SMV 입력 프로그램>

테스트 케이스 변환 모델에 대한 모델체킹(1)

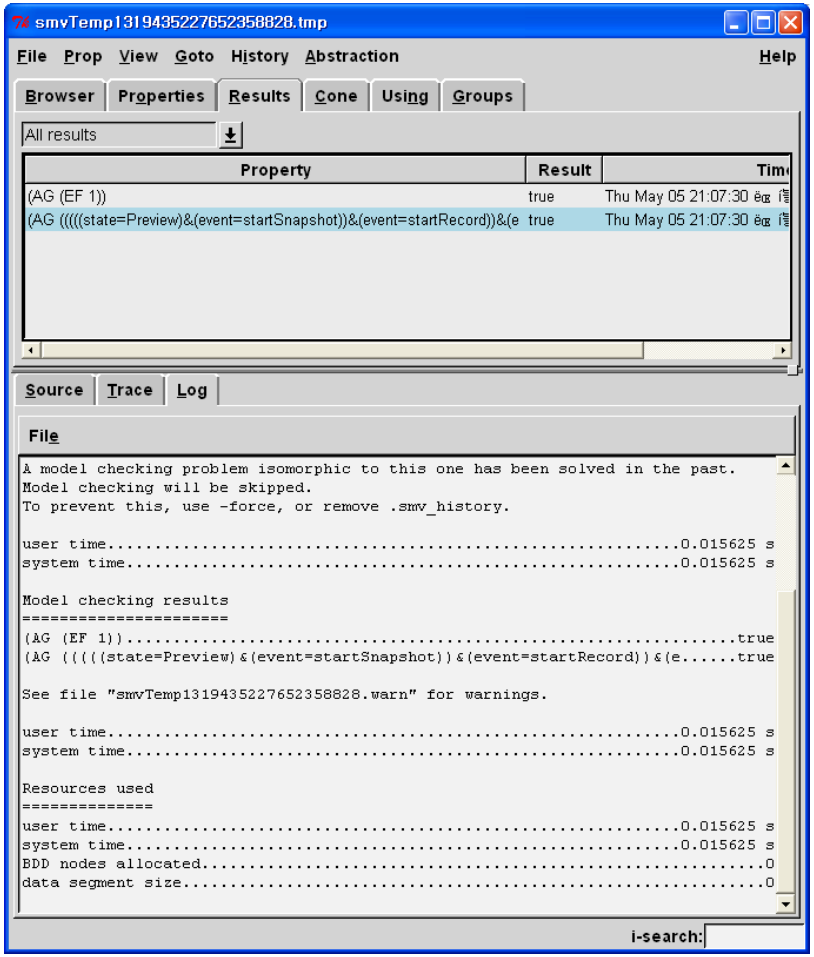


- SPEC AG EF 1;
  - 모델의 모든 state로 접근할 수 있음(reachability)

- SPEC AG (((state=Preview) & (event=startSnapshot) &(event=startRecord) &(event=stopPostview)) ->AX(state=Snapshot));
  - 한번에 여러 개의 이벤트가 발생해도 Snapshot state로 넘어감

    -> 사진 촬영이 수행 됨

테스트 케이스 변환 모델에 대한 모델체킹(2)



- SPEC AG ((event=startSnapshot) ->AX(state=Snapshot));
  - 사진 촬영 명령이 오면 모든 state에서 바로 사진촬영이 이루어 지는지 확인(만족하면 안 되는 속성)

- Counter Example
  - AlbumEditing state에 있을 경우 startSnapshot이 오면 현재 state에 머뭄

  -> AlbumEditing state에서는 사진촬영명령이 왔을 때 반응하지 않음

결론 및 향후 연구

- 결론
  - 테스트 케이스의 안전성 고려여부를 확인 할 수 있는 기법을 제시하였고 간단한 예제에 적용하여 보았다.
  - 고장수목분석 및 정형화되지 않은 테스트 요구사항에 의해 사람이 직접 수행한 과정이 있다.

- 향후 연구
  - 요구사항 뿐만 아니라, 디자인, 소스 코드까지 감안한 다양한 테스트케이스를 정형 모델로 변환하고 이에 대한 모델체킹을 하여 실용성을 높인다.
  - 각 과정을 자동화하는 연구를 계획하고 있다.
    - 이를 위해 각 단계의 산출물 간의 일치성을 확인할 수 있는 방법 및 정형화된 모델을 정의할 필요가 있다.
  - 안전성 뿐만 아니라 신뢰성(reliability), 보안성(security)에 대한 고려를 하는 연구로 확장할 수 있을 것이다.