ML 프로그램에서 예외상황을 발생시키는 테스트 데이타 생성 방법

류석영, 이광근

KAIST 전산학과

요 약

주어진 ML 프로그램에서 예외 상황을 발생시키는 테스트 데이타를 자동으로 생성해주는 분석기를 제안한다. 타입 검사를 거친 ML 프로그램이 비정상적으로 멈추는 경우는, 프로그램이 예외상황을 발생시킬 때 그 예외상황을 적절하게 처리해주지 못한 경우밖에 없다. 따라서 프로그램 내의 모든 예외상황을 발생시키는 테스트 데이타를 만들어서, 각각의 경우에 프로그램이 제대로 처리해주도록 보완하면 안정성(safety)이 보장되는 프로그램을 개발할 수 있게 된다.

주어진 프로그램에서 각각의 예외상황 $\mathtt{E}_i$가 발생한다는 조건을 가지고, 프로그램의 입력 값인 테스트 데이타 집합 D_i 구한다. 전체 프로그램의 값인 예외 상황 $\mathtt{E}_i$를 시작점으로 하여 프로그램 내의 각 내부 식들이 가져야하는 값들을 "강요"해가면서, 그 과정 중에 테스트 데이타의 값을 구하게 된다. 분석 결과로 얻게 되는 테스트 데이타들(D_i)은, 그 중에 어느 값을 입력으로 사용하여 프로그램을 수행시켜도 예외상황 $\mathtt{E}_i$가 발생한다는 것을 보장해야 한다. 이렇게 구한 테스트 데이타 집합 D_i에서 각각 하나씩의 데이타를 선택하면, 프로그램 내의 모든 예외 상황을 발생시키는 테스트 데이타 집합을 구하게 된다.

테스트 데이타를 생성하는 분석 방법을 제안하고, 몇 가지 예를 통하여 이 분석 방법이 만족해야할 성질에 대해서 논의한다.

[현재 진행중인 연구에 대한 초기단계의 리포트]

1. 서론

ML 프로그래머는 예외상황(exception) 메카니즘을 사용하여 프로그램의 흐름을 자유롭게 제어할 수 있다. 프로그래머는 예외상황을 자유롭게 정의하고 발생시킬(raise) 수 있으며, 발생한 예외상황을 다양한 방법으로 처리할 수 있다. 예외상황이 발생하면 프로그램의 흐름은 그 예외상황을 처리해주는(handle) 부분으로 전달된다.

그러나 프로그래머가 예외상황을 주의깊게 사용하지 않으면, ML 프로그램의 안전성(safety)을 보장할 수 없게 된다. 타입 검사를 거친 ML 프로그램이 비정상적으로 실행을 멈추는 경우는, 프로그램 내에서 예외상황을 발생시킨 후 적절하게 처리해주지 않은 경우밖에 없다.

따라서 프로그램에서 처리되지 않는 예외상황을 모두 적절하게 처리해주면, 안전성이 보장되는 프로그램을 개발할 수 있게 된다. ML 프로그램의 모든 예외상황을 발생시키는 테스트 데이타를 만들어서, 그 테스트 데이타를 이용하여 각각의 예외상황을 발생시킨 후, 프로그램이 제대로 처리하지 못하는 예외상황들을 찾아낸다. 이러한 예외상황들을 적절하게 처리해주면 그 프로그램의 안전성은 보장되는 것이다.

주어진 ML 프로그램에서 예외상황을 발생시키는 테스트 데이타를 자동으로 생성해주는 분석기를 제안한다. 몇 가지 예를 통하여 이 분석 방법이 만족해야 하는 성질과 앞으로 더 필요한 일에 대하여 논의한다.

2. 언어

분석 대상으로 하는 언어는 ML 언어 [MTHM97]의 일부분이다. 언어의 문법구조는 다음과 같다.

e	::=	$\mathtt{0}$	상수
		x	변수
		$\lambda x.e$	함수
		$\mathtt{fix}\ f\ \lambda x.e_1\ \mathtt{in}\ e_2$	재귀 함수
		$e_1\ e_2$	함수 호출
		$\kappa\ e$	데이타 생성
		$\mathtt{case}\ e_1\ \kappa(x)\ e_2\ e_3$	분기
		$\mathtt{raise}\ \mathtt{E}$	예외상황 발생
		$\mathtt{handle}\ e_1\ \mathtt{E}\ \lambda x.e_2$	예외상황 처리

"$\mathtt{fix}\ f\ \lambda x.e_1\ \mathtt{in}\ e_2$"는 e_2 안에서 f를 $\lambda x.e_1$로 바인드시켜준다. "$\kappa\ e$"는 e를 계산한 값이 v일 때, $\kappa\ v$가 된다. "$\mathtt{case}\ e_1\ \kappa(x)\ e_2\ e_3$"는 e_1의 값이 $\kappa\ v$이면 v를 x에 바인드시키고 전체 식의 값은 e_2의 값이 되고, 그렇지 않으면 전체 식의 값이 e_3의 값이 된다. 이 언어의 "raise" 표현식은 단지 하나의 예외상황 E만을 발생시킬 수 있다. 그러나 ML 프로그램의 "$\mathtt{raise}\ e$"는 저자들의 예외상황 분석 방법 *ExnAnal* [YR97]을 이용하여, $\mathit{ExnAnal}(e) \supseteq \{\mathtt{E}\}$인 경우에 "$\mathtt{raise}\ \mathtt{E}$"으로 변환할 수 있다. "$\mathtt{handle}\ e_1\ \mathtt{E}\ \lambda x.e_2$"는, e_1을 계산해서 예외상황 E가 발생하면 그 예외상황을 x에 바인드시켜서 e_2를 계산한 값을 갖고, 그렇지 않으면 e_1의 값을 갖는다. 이 언어의 "handle" 표현식도 단지 하나의 예외상황 E만을 처리할 수 있다. 그러나 "handle" 표현식을 반복해서 사용하여 "$\mathtt{handle}\ (\mathtt{handle}\ e_1\ \mathtt{E}\ \lambda x.e_2)\ \mathtt{F}\ \lambda y.e_3$"와 같이 확장할 수 있다.

3. 테스트 데이타 생성 분석 방법

주어진 ML 프로그램에서 예외상황을 발생시키는 테스트 데이타를 생성하는 분석 방법을 제안한다. 프로그램 내의 모든 예외상황 $\{\mathtt{E}_1, \cdots, \mathtt{E}_N\}$에 대하여, 각각의 예외상황 $\mathtt{E}_i$가 발생한다는 조건을 가지고 프로그램의 입력 값인 테스트 데이타 집합 D_i 구한다. 이렇게 구한 테스트 데이타 집합 D_i에서 각각 하나씩의 데이타를 선택하면, 프로그램 내의 모든 예외 상황을 발생시키는 테스트 데이타 집합을 구하게 된다.

각각의 예외상황 $\mathtt{E}_i$에 대하여, 프로그램에서 예외 상황 $\mathtt{E}_i$가 발생한다는 것을 시작점으로 하여 프로그램 내의 각 내부 식들이 가져야하는 값들을 "강요"해간다. 이 과정 중에 프로그램 내의 자유 변수(free variable)인 테스트 데이타의 값을 구하게 된다. 분석 결과로 얻게 되는 테스트 데이타들(D_i)은, 그 중에 어느 값을 입력으로 사용하여 프로그램을 수행시켜도 예외상황 $\mathtt{E}_i$가 발생한다는 것을 보장해야 한다.

이 분석 방법은 가능한 모든 실행 경로를 다 포함해야 하는 분석 방법 (upper approximation)과 달리, 예외상황을 발생시키는 실행 경로를 하나라도 포함하기만 하면(lower approximation) 된다. 따라서, 발생할 수 있는 모든 경우를 다 포함할 필요는 없지만, 잘못된 경우를 포함하면 안된다.

테스트 데이타 생성 분석의 규칙과 이 분석이 만족해야 하는 성질, 몇 가지 예를 들기로 한다.

3.1 집합 제약식(Set Constraints)

테스트 데이타 생성 분석은 다음의 집합 제약식을 이용해서 표현한다.

3.1.1 문법구조(Syntax)

테스트 데이타 생성 규칙은 "$u \triangleright e : \mathcal{D}$"으로 표현한다. e는 프로그램 식(expression)을 나타내고, u와 $\mathcal{D}$는 각각 다음과 같이 정의된 집합 표현식(set expression), 집합 제약식을 나타낸다.

$$
\begin{array}{rcl}
u & ::= & \mathcal{X} \\
 & | & 0 \\
 & | & \kappa\,\mathcal{X} \\
 & | & \kappa^{-1}\,\mathcal{X} \\
 & | & \bar{\kappa}\,\mathcal{X} \\
 & | & \lambda x.e \\
 & | & \hat{\mathtt{E}}
\end{array}
\qquad
\begin{array}{rcl}
\mathcal{D} & ::= & \mathcal{C}_1 \vee \mathcal{C}_2 \\
 & | & \mathcal{C}
\end{array}
\qquad
\begin{array}{rcl}
\mathcal{C} & ::= & \mathcal{X} \subseteq u \\
 & | & \kappa\,\mathcal{Y} \subseteq \mathcal{X} \\
 & | & \lambda x.e \subseteq \mathcal{X} \\
 & | & \mathcal{C}_1 \wedge \mathcal{C}_2 \\
 & | & \bot
\end{array}
$$

$\mathcal{D}$는 만족해야하는 집합 제약식들의 모임(conjunctions)의 선택적인 모임(disjunction)이다. $\vee$로 연결된 모든 식들은 그 내부에 $\vee$를 포함하지 않아야 한다. 즉, DNF(disjunctive normal form)이어야 한다. 분석 규칙을 적용할 때마다 이 조건을 만족하도록 하기 위해서 $\dot{\wedge}$ 연산자를 정의한다.

$\mathcal{D}_1 \equiv \mathcal{D}'_1 \vee \mathcal{C}_1 \equiv (\mathcal{D}' \vee \mathcal{C}') \vee \mathcal{C}_1$이고 $\mathcal{D}_2 \equiv \mathcal{D}'_2 \vee \mathcal{C}_2 \equiv (\mathcal{D}'' \vee \mathcal{C}'') \vee \mathcal{C}_2$라 하면, $\mathcal{D}_1 \dot{\wedge} \mathcal{D}_2$는 다음과 같이 정의된다.

$$
\begin{array}{rcl}
\mathcal{D}_1 \dot{\wedge} \mathcal{D}_2 & = & (\mathcal{D}'_1 \vee \mathcal{C}_1) \dot{\wedge} (\mathcal{D}'_2 \vee \mathcal{C}_2) \\
 & = & (\mathcal{D}'_1 \dot{\wedge} \mathcal{D}'_2) \vee (\mathcal{D}'_1 \dot{\wedge} \mathcal{C}_2) \vee (\mathcal{C}_1 \dot{\wedge} \mathcal{D}'_2) \vee (\mathcal{C}_1 \wedge \mathcal{C}_2) \\
\mathcal{D}'_1 \dot{\wedge} \mathcal{C}_2 & = & (\mathcal{D}' \vee \mathcal{C}') \dot{\wedge} \mathcal{C}_2 \\
 & = & (\mathcal{D}' \dot{\wedge} \mathcal{C}_2) \vee (\mathcal{C}' \wedge \mathcal{C}_2) \\
\mathcal{C}_1 \dot{\wedge} \mathcal{D}'_2 & = & \mathcal{C}_1 \dot{\wedge} (\mathcal{D}'' \vee \mathcal{C}'') \\
 & = & (\mathcal{C}_1 \dot{\wedge} \mathcal{D}'') \vee (\mathcal{C}_1 \wedge \mathcal{C}'')
\end{array}
$$

3.1.2 의미구조(Semantics)

테스트 데이타 생성 규칙 "$u \triangleright e : \mathcal{D}$"의 의미는, 프로그램 식 e를 계산한 값이 $\mathcal{I}(u)$에 포함되려면, $\mathcal{I}(\mathcal{D})$를 만족해야 한다는 것이다.

각 집합 제약식과 집합 표현식의 의미는 다음과 같이 정의된다.

$$
\begin{array}{rcll}
v & \in & \mathit{Val} & = \mathit{Closure} + \mathit{Constant} + \mathit{Exn} + \{0\} \\
\lambda x.e & \in & \mathit{Closure} & = \mathit{Expr} \\
\kappa\, v & \in & \mathit{Constant} & = \mathit{Con} \times \mathit{Val} \\
\kappa & \in & \mathit{Con} & = \{\kappa_1, \cdots, \kappa_N\} \\
\hat{\mathtt{E}} & \in & \mathit{Exn} & = \{\hat{\mathtt{E}}_1, \cdots, \hat{\mathtt{E}}_N\}
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{I}(\mathcal{X}) & \subseteq & \mathit{Val} \\
\mathcal{I}(\mathtt{0}) & = & \{\mathtt{0}\} \\
\mathcal{I}(\kappa\ \mathcal{X}) & = & \{\kappa\, v \mid v \in \mathcal{I}(\mathcal{X})\} \\
\mathcal{I}(\kappa^{-1}\ \mathcal{X}) & = & \{v \mid \kappa\, v \in \mathcal{I}(\mathcal{X})\} \\
\mathcal{I}(\bar{\kappa}\ \mathcal{X}) & = & \{\kappa'\, v \mid v \in \mathcal{I}(\mathcal{X}),\ \kappa' \neq \kappa\} \\
\mathcal{I}(\lambda x.e) & = & \{\lambda x.e\} \\
\mathcal{I}(\hat{\mathtt{E}}) & = & \{\hat{\mathtt{E}}\} \\
 & & \\
\mathcal{I}(\mathcal{C}_1 \vee \mathcal{C}_2) & = & \mathcal{I}(\mathcal{C}_1) \text{ or } \mathcal{I}(\mathcal{C}_2) \\
 & & \\
\mathcal{I}(\mathcal{X} \subseteq u) & = & \mathcal{I}(\mathcal{X}) \ \subseteq\ \mathcal{I}(u) \\
\mathcal{I}(\kappa\ \mathcal{Y} \subseteq \mathcal{X}) & = & \mathcal{I}(\kappa\ \mathcal{Y}) \ \subseteq\ \mathcal{I}(\mathcal{X}) \\
\mathcal{I}(\lambda x.e \subseteq \mathcal{X}) & = & \mathcal{I}(\lambda x.e) \ \subseteq\ \mathcal{I}(\mathcal{X}) \\
\mathcal{I}(\mathcal{C}_1 \wedge \mathcal{C}_2) & & \\
\boxed{\begin{array}{lcl}
\mathcal{I}((\mathcal{X} \subseteq u_1) \wedge (\mathcal{X} \subseteq u_2)) & = & \mathcal{I}(\mathcal{X}) \subseteq (\mathcal{I}(u_1) \cap \mathcal{I}(u_2)) \\
\mathcal{I}((u_1 \subseteq \mathcal{X}) \wedge (\mathcal{X} \subseteq u_2)) & = & \mathcal{I}(u_1) \subseteq \mathcal{I}(\mathcal{X}) \subseteq \mathcal{I}(u_2) \\
\mathcal{I}((u_1 \subseteq \mathcal{X}) \wedge (u_2 \subseteq \mathcal{X})) & = & (\mathcal{I}(u_1) \cup \mathcal{I}(u_2)) \subseteq \mathcal{I}(\mathcal{X})
\end{array}} & & \\
\mathcal{I}(\bot) & = & \emptyset
\end{array}
$$

프로그램 내의 모든 변수 x는 집합 변수(set variable) $\mathcal{X}$로 표현된다. E는 발생되지 않은 예외상황(exception constuctor)를 의미하고, $\hat{\mathtt{E}}$은 발생된 예외상황(exception packet)을 의미한다. "$\mathcal{I}(\mathcal{C}_1)$ or $\mathcal{I}(\mathcal{C}_2)$"는 $\mathcal{I}(\mathcal{C}_1)$과 $\mathcal{I}(\mathcal{C}_2)$ 중 어느 모델을 선택해도 된다는 것을 의미한다. $\mathcal{I}(\mathcal{C}_1 \wedge \mathcal{C}_2)$의 의미는 가능한 경우에 대해서 네모 안에 정의하였다.

$\bot$은 집합 제약식들이 모순이 되는 경우를 나타낸다. 즉, $\bot$을 만족하는 모델은 존재하지 않는다. 따라서, $\bot$과 다른 집합 제약식과의 $\wedge$은 $\bot$이 되고 ($\mathcal{C} \wedge \bot = \bot \wedge \mathcal{C} = \bot$), $\bot$과 다른 집합 제약식과의 $\vee$은 그 집합 제약식이 된다 ($\bot = \bot \vee \mathcal{C} = \mathcal{C}$).

3.2 분석 규칙

테스트 데이타 생성 분석의 규칙 중 기본적인 규칙들은 그림 1과 같다. $u \rhd e : \mathcal{D}$에서 e의 값이 u에 포함되도록 조건을 만들어주는 규칙이다.

[C1]과 같이, e의 값이 u에 포함되는 경우에는 이 관계식 (relation)이 항상 만족하므로 $\mathcal{D}$에 아무런 조건도 만들어주지 않는다. [C2]와 같이, u가 집합 변수인 경우에는 e의 값이 그 집합 변수에 포함되도록 $\{\mathtt{0} \subseteq \mathcal{X}\}$ 조건을 만들어준다. [C3]와 같이, u의 값이 $\kappa^{-1}\ \mathcal{X}$인 경우에는 $\kappa\ e$를 집합 변수 $\mathcal{X}$에 포함되도록 $\{\kappa\ \mathtt{0} \subseteq \mathcal{X}\}$ 조건을 만들어준다. [C4]와 같이, e의 값이 u에 포함될 수 없는 경우에는 모순이 생기므로 $\bot$을 만들어준다.

[NV1]과 [NV2]는 함수 이름이 아닌 일반 변수에 대한 규칙이다. 변수 x의 값이 발생한 예외상황(exception packet)이 되어야 한다는 조건은 모순이 생기고 ([NV2]), 그 이외의 경우에는 변수 x가 u의 값에 포함되도록 조건을 만들어준다 ([NV1]).

그 외의 규칙들은 그림 2, 그림 3과 같다.

[CON4]의 변수 $\mathcal{Y}$는 $\kappa^{-1}\mathcal{X}$를 대신해서 사용한 변수이다. $\kappa'^{-1}(\kappa^{-1}\mathcal{X})$ 대신 $\kappa^{-1}\mathcal{Y}$를 사용하여, u의 문법 구조에서 κ^{-1}의 인수로 집합 변수만 오도록 한 조건을 만족하도록 만든다. 또한, $\dot{\wedge}$를 사용하여, $\mathcal{D}$가 DNF를 유지하도록 만든다.

다른 경우는 [CASE2]를 이용하여 설명하기로 한다. case 표현식에서 예외상황이 발생했다면, e_1이나 e_2, e_3 중의 한 표현식에서 예외상황이 발생했을 것이다. 따라서 [CASE2]에서는 이 세 가지 경우에 대하여 분석을 한 후, 그 결과를 $\vee$으로

$$
\begin{array}{lrcll}
\texttt{[C1]} & \texttt{0} & \rhd & \texttt{0} : \emptyset & \\
\texttt{[C2]} & \mathcal{X} & \rhd & \texttt{0} : \{\texttt{0} \subseteq \mathcal{X}\} & \\
\texttt{[C3]} & \kappa^{-1}\ \mathcal{X} & \rhd & \texttt{0} : \{\kappa\ \texttt{0} \subseteq \mathcal{X}\} & \\
\texttt{[C4]} & v & \rhd & \texttt{0} : \bot & \text{where } v \in \{\kappa\ \mathcal{X},\ \bar{\kappa}\ \mathcal{X},\ \lambda x.e,\ \hat{\texttt{E}}\} \\
 & & & & \\
\texttt{[NV1]} & v & \rhd & x : \{\mathcal{X} \subseteq v\} & \text{where } v \in u \setminus \{\hat{\texttt{E}}\} \\
\texttt{[NV2]} & \hat{\texttt{E}} & \rhd & x : \bot & \\
 & & & & \\
\texttt{[FnV1]} & \lambda x.e & \rhd & f : \emptyset & \text{where } \texttt{fix}\ f\ \lambda x.e \in e_0 \\
\texttt{[FnV2]} & \mathcal{Y} & \rhd & f : \{\lambda x.e \subseteq \mathcal{Y}\} & \text{where } \texttt{fix}\ f\ \lambda x.e \in e_0 \\
\texttt{[FnV3]} & \kappa^{-1}\ \mathcal{Y} & \rhd & f : \{\kappa\ (\lambda x.e) \subseteq \mathcal{Y}\} & \text{where } \texttt{fix}\ f\ \lambda x.e \in e_0 \\
\texttt{[FnV4]} & v & \rhd & f : \bot & \text{where } \texttt{fix}\ f\ \lambda x.e \in e_0 \\
 & & & & \quad \text{and } v \in \{\texttt{0},\ \kappa\ \mathcal{X},\ \bar{\kappa}\ \mathcal{X},\ \lambda y.e' \neq \lambda x.e,\ \hat{\texttt{E}}\} \\
\texttt{[ABS1]} & \lambda x.e & \rhd & \lambda x.e : \emptyset & \\
\texttt{[ABS2]} & \mathcal{Y} & \rhd & \lambda x.e : \{\lambda x.e \subseteq \mathcal{Y}\} & \\
\texttt{[ABS3]} & \kappa^{-1}\ \mathcal{Y} & \rhd & \lambda x.e : \{\kappa\ (\lambda x.e) \subseteq \mathcal{Y}\} & \\
\texttt{[ABS4]} & v & \rhd & \lambda x.e : \bot & \text{where } v \in \{\texttt{0},\ \kappa\ \mathcal{X},\ \bar{\kappa}\ \mathcal{X},\ \lambda y.e' \neq \lambda x.e,\ \hat{\texttt{E}}\} \\
 & & & & \\
\texttt{[RS1]} & \hat{\texttt{E}} & \rhd & \texttt{raise E} : \emptyset & \\
\texttt{[RS2]} & v & \rhd & \texttt{raise E} : \bot & \text{where } v \in \{\mathcal{X},\ \texttt{0},\ \kappa\ \mathcal{X},\ \kappa^{-1}\ \mathcal{X},\ \bar{\kappa}\ \mathcal{X},\ \lambda x.e,\ \hat{\texttt{E}}' \neq \hat{\texttt{E}}\} \\
 & & & & \\
\texttt{[CON1]} & v & \rhd & \kappa\ e : \bot & \text{where } v \in \{\texttt{0},\ \bar{\kappa}\ \mathcal{X},\ \lambda x.e\}
\end{array}
$$

그림 1: 기본적인 규칙들

모은다. 즉, e_1에서 예외상황이 발생한 경우의 분석 결과인 $\mathcal{D}_1$, e_2에서 예외상황이 발생한 경우인 $\mathcal{D}_2'$과 이 때 e_1이 $\kappa\ \mathcal{X}$라는 조건으로 분석한 $\mathcal{D}_1'$의 $\wedge$, 마지막으로 e_3에서 예외상황이 발생한 경우인 $\mathcal{D}_2''$과 이 때 e_1이 $\bar{\kappa}\ \mathcal{Y}$라는 조건으로 분석한 $\mathcal{D}_1''$의 $\wedge$을 $\vee$로 모은다.

$$
\texttt{[CON2]}\quad \frac{\kappa^{-1}\ \mathcal{X} \rhd e : \mathcal{D}}{\mathcal{X} \rhd \kappa\ e : \mathcal{D}} \qquad \texttt{[CON3]}\quad \frac{\mathcal{X} \rhd e : \mathcal{D}}{\kappa\ \mathcal{X} \rhd \kappa\ e : \mathcal{D}}
$$

$$
\texttt{[CON4]}\quad \frac{\kappa'^{-1}\ \mathcal{Y} \rhd e : \mathcal{D}}{\kappa^{-1}\ \mathcal{X} \rhd \kappa'\ e : \mathcal{D} \dot{\wedge} \{\mathcal{Y} \subseteq \kappa^{-1}\ \mathcal{X}\}} \quad \text{fresh } \mathcal{Y}
$$

$$
\texttt{[CON5]}\quad \frac{\hat{\texttt{E}} \rhd e : \mathcal{D}}{\hat{\texttt{E}} \rhd \kappa\ e : \mathcal{D}}
$$

그림 2: 그 외의 규칙들 (I)

[FIX] $$\frac{u \rhd e_2 \,:\, \mathcal{D}}{u \rhd \texttt{fix}\ f\ \lambda x.e_1\ \texttt{in}\ e_2 \,:\, \mathcal{D}}$$

[APP1] $$\frac{\begin{array}{c} Lam(e_1) = \lambda x.e' \qquad v \in u \setminus \{\hat{\mathtt{E}}\} \\ \lambda x.e' \rhd e_1 \,:\, \mathcal{D}_1 \qquad \mathcal{X} \rhd e_2 \,:\, \mathcal{D}_2 \qquad v \rhd e' \,:\, \mathcal{D}_3 \end{array}}{v \rhd e_1\ e_2 \,:\, \mathcal{D}_1 \dot{\wedge} \mathcal{D}_2 \dot{\wedge} \mathcal{D}_3}$$

[APP2] $$\frac{\begin{array}{c} \hat{\mathtt{E}} \rhd e_1 \,:\, \mathcal{D}_1 \\ \lambda x.e' \rhd e_1 \,:\, \mathcal{D}'_1 \qquad \hat{\mathtt{E}} \rhd e_2 \,:\, \mathcal{D}'_2 \\ \lambda x.e' \rhd e_1 \,:\, \mathcal{D}''_1 \qquad \mathcal{X} \rhd e_2 \,:\, \mathcal{D}''_2 \qquad \hat{\mathtt{E}} \rhd e' \,:\, \mathcal{D}''_3 \end{array}}{\hat{\mathtt{E}} \rhd e_1\ e_2 \,:\, \mathcal{D}_1 \vee (\mathcal{D}'_1 \dot{\wedge} \mathcal{D}'_2) \vee (\mathcal{D}''_1 \dot{\wedge} \mathcal{D}''_2 \dot{\wedge} \mathcal{D}''_3)} \quad Lam(e_1) = \lambda x.e'$$

[CASE1] $$\frac{\begin{array}{cc} \kappa\ \mathcal{X} \rhd e_1 \,:\, \mathcal{D}_1 & v \rhd e_2 \,:\, \mathcal{D}_2 \\ \bar{\kappa}\ \mathcal{Y} \rhd e_1 \,:\, \mathcal{D}'_1 & v \rhd e_3 \,:\, \mathcal{D}'_2 \end{array}}{v \rhd \texttt{case}\ e_1\ \kappa(x)\ e_2\ e_3 \,:\, (\mathcal{D}_1 \dot{\wedge} \mathcal{D}_2) \vee (\mathcal{D}'_1 \dot{\wedge} \mathcal{D}'_2)} \quad \begin{array}{c} v \in u \setminus \{\hat{\mathtt{E}}\} \\ \text{fresh } \mathcal{Y} \end{array}$$

[CASE2] $$\frac{\begin{array}{cc} \hat{\mathtt{E}} \rhd e_1 \,:\, \mathcal{D}_1 & \\ \kappa\ \mathcal{X} \rhd e_1 \,:\, \mathcal{D}'_1 & \hat{\mathtt{E}} \rhd e_2 \,:\, \mathcal{D}'_2 \\ \bar{\kappa}\ \mathcal{Y} \rhd e_1 \,:\, \mathcal{D}''_1 & \hat{\mathtt{E}} \rhd e_3 \,:\, \mathcal{D}''_2 \end{array}}{\hat{\mathtt{E}} \rhd \texttt{case}\ e_1\ \kappa(x)\ e_2\ e_3 \,:\, \mathcal{D}_1 \vee (\mathcal{D}'_1 \dot{\wedge} \mathcal{D}'_2) \vee (\mathcal{D}''_1 \dot{\wedge} \mathcal{D}''_2)} \quad \text{fresh } \mathcal{Y}$$

[HNDL1] $$\frac{\begin{array}{cc} v \rhd e_1 \,:\, \mathcal{D}_1 & \\ \hat{\mathtt{E}} \rhd e_1 \,:\, \mathcal{D}'_1 & v \rhd e_2 \,:\, \mathcal{D}'_2 \end{array}}{v \rhd \texttt{handle}\ e_1\ \mathtt{E}\ \lambda x.e_2 \,:\, \mathcal{D}_1 \vee (\mathcal{D}'_1 \dot{\wedge} \mathcal{D}'_2 \dot{\wedge} \{\hat{\mathtt{E}} \subseteq \mathcal{X}\})} \quad v \in u \setminus \{\hat{\mathtt{E}},\ \hat{\mathtt{E}}'\}$$

[HNDL2] $$\frac{\begin{array}{c} \hat{\mathtt{E}} \rhd e_1 \,:\, \mathcal{D}_1 \\ \hat{\mathtt{E}}' \rhd e_1 \,:\, \mathcal{D}'_1 \qquad \hat{\mathtt{E}} \rhd e_2 \,:\, \mathcal{D}'_2 \end{array}}{\hat{\mathtt{E}} \rhd \texttt{handle}\ e_1\ \mathtt{E}'\ \lambda x.e_2 \,:\, \mathcal{D}_1 \vee (\mathcal{D}'_1 \dot{\wedge} \mathcal{D}'_2 \dot{\wedge} \{\hat{\mathtt{E}} \subseteq \mathcal{X}\})}$$

[HNDL3] $$\frac{\hat{\mathtt{E}} \rhd e_1 \,:\, \mathcal{D}_1 \qquad \hat{\mathtt{E}} \rhd e_2 \,:\, \mathcal{D}_2}{\hat{\mathtt{E}} \rhd \texttt{handle}\ e_1\ \mathtt{E}\ \lambda x.e_2 \,:\, \mathcal{D}_1 \dot{\wedge} \mathcal{D}_2 \dot{\wedge} \{\hat{\mathtt{E}} \subseteq \mathcal{X}\}}$$

그림 3: 그 외의 규칙들 (II)

3.3 안전성(Safety)

테스트 데이타 생성 분석은 발생할 수 있는 모든 경우를 다 포함할 필요는 없지만, 잘못된 경우를 포함하면 안된다. 즉, 분석 결과로 얻은 테스트 데이타를 입력값으로 하여 프로그램을 실행하면, 예외상황 E가 발생해야 한다. 이 성질은 다음과 같이 정의할 수 있다.

정리 1. *안전성(Safety)*
$\hat{\mathtt{E}} \rhd e_0 : \mathcal{D},\ FV(e_0) = \{x\},\ \mathcal{D} \equiv \mathcal{C}_1 \vee \cdots \vee \mathcal{C}_n,\ lm(\mathcal{C}_i)(x) = \{v_1, \cdots, v_m\},$
then $\forall\ v_j \in lm(\mathcal{C}_i)(x).\ [v_j/x]\ e_0 \overset{*}{\longmapsto} \hat{\mathtt{E}}$.

분석 결과로 얻은 집합 제약식들의 모임(disjunction) ($\mathcal{D}$) 중에서 임의의 하나 ($\mathcal{C}_i$)를 택하여 그 모델을 구하면, 그 모델에서 얻을 수 있는 테스트 데이타 중 어느 것 (v_j)을 입력 값 (x)으로 하여 프로그램 (e_0)을 실행해도 예외상황 E가 발생해야 한다. 이 성질을 증명해야, 제안한 분석 방법이 옳다는 것을 확인할 수 있다.

3.4 예

3.4.1 간단한 예

$$\begin{array}{ll} \mathtt{fix} & f\ \lambda z.\ \overbrace{\mathtt{case}\ z\ (\mathtt{Zero}\ y)\ (\mathtt{raise\ E})\ (\mathtt{Suc(Zero\ 0)})}^{e_1} \\ \mathtt{in} & (f\ x) \end{array}$$

간단한 예로서, 위의 프로그램을 분석하는 경우를 설명하기로 한다. [FIX] 규칙에 의해서, in 다음에 오는 $(f\ x)$만을 분석하면 된다.

$$\frac{\begin{array}{c} Lam(f) = \lambda z.e_1 \\ \hat{\mathtt{E}} \rhd f : \bot \\ \lambda z.e_1 \rhd f : \emptyset \qquad \hat{\mathtt{E}} \rhd x : \bot \\ \lambda z.e_1 \rhd f : \emptyset \qquad \mathcal{Z} \rhd x : \{\mathcal{X} \subseteq \mathcal{Z}\} \qquad \hat{\mathtt{E}} \rhd e_1 : \mathcal{D} \end{array}}{\hat{\mathtt{E}} \rhd f\ x : \bot \vee \bot \vee (\{\mathcal{X} \subseteq \mathcal{Z}\} \dot{\wedge} \mathcal{D})}$$

[APP2] 규칙에 의해서, $(f\ x)$의 분석은 위와 같이 세 가지 경우를 모으면(disjunction) 된다. f에서 예외상황이 발생하는 경우와 x에서 예외상황이 발생하는 경우는 모순이 생겨서 제외하고, f의 내용(body)인 e_1에서 예외상황이 발생하는 경우를 분석하면 다음과 같이 된다.

$$\frac{\begin{array}{c} \hat{\mathtt{E}} \rhd z : \bot \\ \mathtt{Zero}\ \mathcal{Y} \rhd z : \{\mathcal{Z} \subseteq \mathtt{Zero}\ \mathcal{Y}\} \qquad \hat{\mathtt{E}} \rhd \mathtt{raise\ E} : \emptyset \\ \overline{\mathtt{Zero}}\ \mathcal{W} \rhd z : \{\mathcal{Z} \subseteq \overline{\mathtt{Zero}}\ \mathcal{W}\} \qquad \hat{\mathtt{E}} \rhd \mathtt{Suc(Zero\ 0)} : \bot \end{array}}{\hat{\mathtt{E}} \rhd \mathtt{case}\ z\ (\mathtt{Zero}\ y)\ (\mathtt{raise\ E})\ (\mathtt{Suc(Zero\ 0)}) : \mathcal{D}}$$

$$\text{where } \mathcal{D} \equiv \bot \vee \{\mathcal{Z} \subseteq \mathtt{Zero}\ \mathcal{Y}\} \vee (\{\mathcal{Z} \subseteq \overline{\mathtt{Zero}}\ \mathcal{W}\} \wedge \bot)$$

[CASE2] 규칙에 의해서, e_1의 분석은 위와 같은 세 경우를 모으면 된다. z에서 예외상황이 발생하는 경우와 Suc Zero 0에서 예외상황이 발생하는 경우는 모순이 생겨서 제외하고, raise E에서 예외상황이 발생하므로 답을 찾게 된다.

모순이 생기지 않은 집합 제약식들만을 모아서 답을 구하면 된다. $\mathcal{D}$는 $\{\mathcal{Z} \subseteq \mathtt{Zero}\ \mathcal{Y}\}$가 되므로, $f\ x$를 분석한 결과는 $\{\mathcal{X} \subseteq \mathcal{Z}\} \wedge \{\mathcal{Z} \subseteq \mathtt{Zero}\ \mathcal{Y}\}$가 된다. 따라서 구하고자 하는 테스트 데이타는 x는 Zero u가 된다.

3.4.2 재귀 함수의 예

```
fix  f λx. case x (Zero y) (raise E) (case x (Suc z) (f z) 0)
 in  f (Suc(Suc(Zero 0)))
```

위의 예는 분석이 무한히 진행되어 끝나지 않는 경우이다. 위의 예와 같이 자기 자신을 호출하는 재귀 함수의 경우에는, 함수 자신(body)을 분석할 때 자기 자신을 다시 반복해서 분석하기 때문에, 분석이 무한하게 진행되어 끝나지 않을 수 있게 된다.

4. 논의 사항

- 이 분석 방법의 안전성(정리 1)은 현재 증명 중이다.
- 분석 규칙 중에 사용되었던 함수 흐름 분석(closure analysis) *Lam*은 *upper approximation* 이 아니라 *lower approximation* 이다. 이러한 조건을 만족하는 *Lam*을 제안해야 한다.
- 이 분석 방법은 일종의 *Collecting Semantics* 라고 할 수 있다. 재귀 함수의 예에서 볼 수 있듯이, 분석이 무한하게 진행되어 끝나지 않는 경우가 발생한다. 따라서 실제적으로 사용될 수 있을 만큼의 정확도와 비용을 갖도록 하는 요약(abstraction) 방법이 필요하다.
- 위의 문제들을 해결한 후에는 이 분석 방법을 SML/NJ와 같이 실제적인 컴파일러에 구현하여, 벤치마크 프로그램을 대상으로 실험해보아야 한다.

참고 자료

[MTHM97] Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML (Revised)*. MIT Press, 1997.

[YR97] Kwangkeun Yi and Sukyoung Ryu. Towards a cost-effective estimation of uncaught exceptions in SML programs. In *Lecture Notes in Computer Science*, volume 1302, pages 98–113. Springer-Verlag, proceedings of the 4th international static analysis symposium edition, 1997.