가변성을 갖는 객체지향언어의 정적 의미를 타입 문맥으로 형식화

Formalization of the Static Semantics of Object-Orientated Languages with Variance via Type Context

홍재민 류석영
PLRG@KAIST

1 서론

여러 객체지향언어는 제네릭과 가변성을 지원한다. 제네릭은 타입 매개변수를 통해 일반화된 타입을 정의할 수 있도록 한다. 가변성은 제네릭이 있는 객체지향언어에 추가적인 서브타입 관계를 정의한다.

가변성을 언어에 추가하면, 클래스를 선언할 때 타입 매개변수를 사용할 수 있는 위치가 제한되고, 언어의 타입-안전성을 보장하기 위해서는 이러한 제약을 언어의 정적 의미에 추가해야 한다.

가변성을 가진 객체지향언어의 정적 의미를 엄밀하게 표현하기 위하여 실행 문맥으로부터 아이디어를 얻어 타입 문맥의 개념을 도입한다. 타입 문맥을 이용해 클래스와 메서드의 올바름을 형식화한다.

2 배경

2.1 제네릭 (Generics)

클래스 내부에서 필드 및 메서드의 타입을 타입 매개변수를 사용하여 정의

객체 생성 시에 타입 인자를 사용하여 해당 객체의 타입 매개변수가 결정

Java 1.4

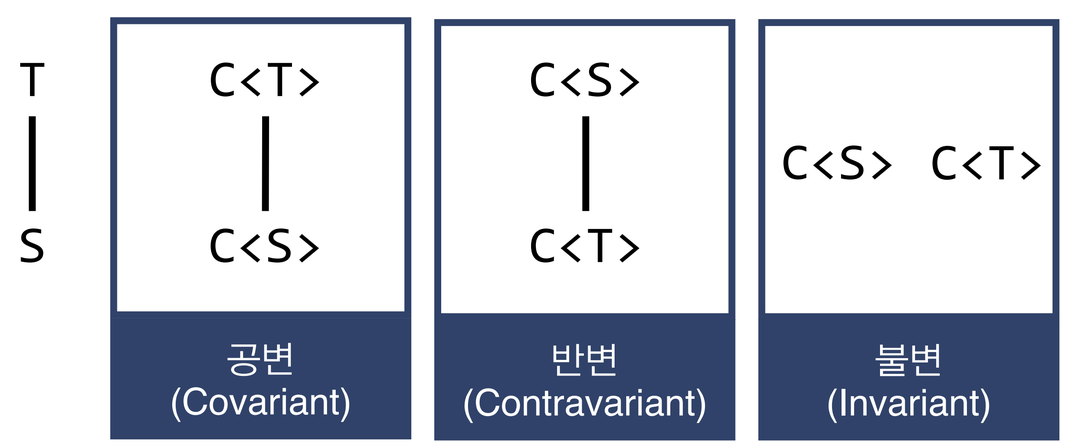
```
class ArrayList extends ... {
    boolean add(Object o) { ... }
    Object get(int index) { ... }
    ...
}
ArrayList al = new ArrayList();
al.add(1);
String str = (String) al.get(0);
```

Java 5

```
class ArrayList<E> extends ... {
    boolean add(E o) { ... }
    E get(int index) { ... }
    …
}
ArrayList<String> al = new ArrayList();
al.add(“abc”);
String str = al.get(0);
```

2.2 가변성 (Variance)

다른 타입 인자로 인스턴스화 된 동일한 제네릭 타입 사이에 추가적인 서브타입 관계 정의




```
<T> List<T> concat(List<T> l, List<T> l0) { ... }
```

2.3 가변성과 클래스의 올바름 (Wellformedness)

클래스의 타입 매개변수를 선언할 때 가변성을 지정 가능

```
sealed abstract class List[+A] extends ... { ... }
```

Scala 2.13

선언처에서 가변성을 지정하면 클래스의 올바름을 판단하는데 새로운 제약이 추가

공변으로 정의된 타입 매개변수는 필드와 메서드 타입에 대해 공변으로만, 반변으로 정의된 타입 매개변수는 반변으로만 사용 가능

```
class MyList[+A] {
    def contains(elem: A): Boolean = ...
    ...
}
val ml: MyList[T] = new MyList[S]() // S <: T
ml.contains(new T())
```

3 타입 문맥

3.1 실행 문맥 (Evaluation Context)

동적 의미를 계산 의미를 통하여 정의하는 데 사용

귀납적으로 정의되며 구멍, 즉 []이 시작점

어떤 표현식(expression)에서 하나의 부분표현식을 구멍으로 대체한 것

$$
\begin{array}{lll}
e & ::= & v \mid e\ e \mid let\ x\ be\ e\ in\ e \\
v & ::= & c \mid x \mid Y \mid \lambda x.e \\
E & ::= & [] \mid E\ e \mid v\ E \mid let\ x\ be\ E\ in\ e
\end{array}
$$

3.2 타입 문맥 (Type Context)

어떤 타입에서 하나의 부품 타입을 구멍으로 대체한 것

타입을 타입 문맥에 적용한 결과는 구멍을 해당 타입으로 대체한 것

타입 문맥 T의 가변성은 T에 타입 매개변수를 적용했을 때 해당 타입 매개변수에 대한 가변성

$$
\begin{array}{lll}
T & ::= & [] \mid (T \Rightarrow t) \mid (t \Rightarrow T) \\
 & \mid & (\overline{t},\ T,\ \overline{t}) \mid O[\overline{t},\ T,\ \overline{t}]
\end{array}
$$

3.3 정적 의미 형식화 (Formalization of Static Semantics)

클래스의 올바름

$$
\frac{\overline{\Delta' \vdash t\ ok} \quad \overline{\Delta'; \text{self} : O[\overline{P}], \overline{x : t}; \overline{V\ P} \vdash \mu\ ok} \quad \overline{(V_i = =) \vee (t_j = T[P_i] \Rightarrow \Delta \vdash T \text{ variance } V_i)}^{i,j} \quad \boxed{\Delta' \vdash c\ ok}}{\Delta \vdash \text{class } O[\overline{V\ P}](\overline{x : t})\ \boxed{\text{extends c}}\ \{\ \overline{\mu}\ \}\ ok}
$$

메서드의 올바름

$$
\frac{distinct(\overline{x}) \quad \overline{\Delta \vdash t\ ok} \quad \Delta \vdash t'\ ok \quad \Gamma \vdash e : t'' \quad \Delta \vdash t'' <: t' \quad \overline{(V_i = =) \vee (((\overline{t}) \Rightarrow t') = T[P_i] \Rightarrow \Delta \vdash T \text{ variance } V_i)}^{i}}{\Delta; \Gamma; \overline{V\ P} \vdash m(\overline{x : t}) : t' \ = \ e\ ok}
$$

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학지원사업 (2016-0-00018) 및 한국연구재단-차세대정보・컴퓨팅기술개발사업 (2017M3C4A7068177)의 지원을 받아 수행된 연구임