분리논리: 프로그램의 안전성에 대한 형식 논증 체계

윤태영(Taeyoung Yoon), 서울대학교 컴퓨터공학과 석박통합과정

2026. 1. 14, KAIST, Korea Logic Day

개론

- 어떻게 프로그램이 옳다는 것을 확인할 수 있을까?
  - 코드 리뷰(Code review), 페어 프로그래밍(Pair programming)
  - 테스팅(Testing): 여러 값을 입력해보고 출력값 확인하기
  - 정적 분석(Static analysis)
  - 퍼징(Fuzzying)
  - 타입 시스템(Type system)
  - **엄밀 검증(Formal verification)**
- 귀납적 방법론 vs. 연역적 방법론
  - 모든 입력값에 대해 원하는 대로 프로그램이 동작한다는 것을 보장
  - 가장 강한 보장을 얻을 수 있음
  - 운영체제(OS), 항공우주 분야, 임베디드 시스템 등 안전이 중요한 프로그램에 활용

발표 순서

1. 호어 논리(Hoare logic) 소개
2. 분리 논리(Separation logic) 소개
3. 예언자 변수(Prophecy variables) 소개
4. 인공지능과의 접점

언어 정의

$$
\begin{array}{lll}
\text{Expressions} & e ::= x \mid v \mid \lambda x.\, e \mid e_1\, e_2 \\
& \quad \mid \textbf{if}\, e\, \textbf{then}\, e_1\, \textbf{else}\, e_2 \\
& \quad \mid e_1 + e_2 \mid e_1 == e_2 \mid e_1 < e_2 \\
& \quad \mid (e_1, e_2) \mid \pi_1\, e \mid \pi_2\, e \\
\text{Values} & v ::= \lambda x.\, e \mid \overline{n} \mid \text{true} \mid \text{false} \mid (v_1, v_2)
\end{array}
$$

$$
\begin{aligned}
(\lambda x.\, e)v &\to e[v/x] \quad \text{(beta reduction)} \\
\textbf{if}\ \text{true}\ \textbf{then}\ e_1\ \textbf{else}\ e_2 &\to e_1 \quad \text{(if-true)} \\
\textbf{if}\ \text{false}\ \textbf{then}\ e_1\ \textbf{else}\ e_2 &\to e_2 \quad \text{(if-false)} \\
\pi_1\,(v_1, v_2) &\to v_1 \quad \text{(proj-fst)} \\
\pi_2\,(v_1, v_2) &\to v_2 \quad \text{(proj-snd)} \\
\overline{n_1} + \overline{n_2} &\to \overline{n_1 + n_2}
\end{aligned}
$$

$$
\frac{n_1 = n_2}{\overline{n_1} == \overline{n_2} \to \text{true}} \qquad \frac{n_1 \neq n_2}{\overline{n_1} == \overline{n_2} \to \text{false}}
$$

$$
\frac{n_1 < n_2}{\overline{n_1} < \overline{n_2} \to \text{true}} \qquad \frac{n_1 \geq n_2}{\overline{n_1} < \overline{n_2} \to \text{false}}
$$

https://tchajed.github.io/sys-verif-fa25/notes/hoare.html#syntax

람다 계산법(lambda calculus) + 불리언 자료형(boolean value) + 쌍(pair)

호어 논리(Hoare logic)와 호어 삼중항(Hoare triple)

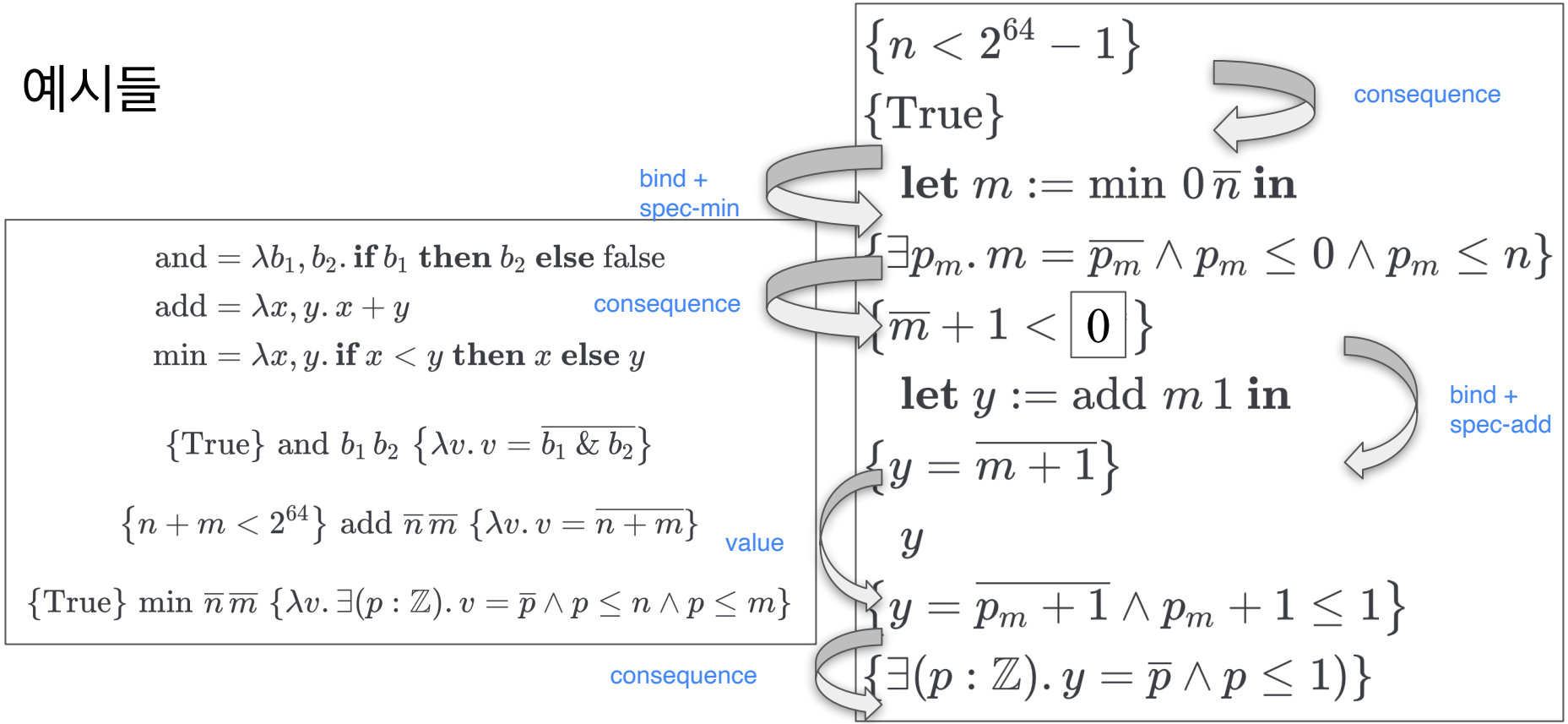
$$\{P\}\ e\ \{\lambda v.Q(v)\}$$

- 의미: P가 성립할 때에 프로그램 e가 만약 값 v로 종료했을 경우, Q(v)가 성립한다
- 전건 P(precondtion)와 후건 Q(postcondition)
- 호어 삼중항은 프로그램의 명세(specification)
  - 명세를 만족하는 여러 프로그램이 있을 수 있음
  - 조립식 논증이 가능함

호어 삼중항의 규칙들

- $\{P(v)\}\ v\ \{\lambda w.P(w)\}$ (value)
  - $\{false=false\}\ false\ \{\lambda w.w = false\}$
- $e1 \rightarrow e2,\ \{P\}\ e2\ \{\lambda w.Q(w)\} \vdash \{P\}\ e1\ \{\lambda w.Q(w)\}$ (pure step)
- $P' \Rightarrow P,\ (\forall v.\ Q(v) \Rightarrow Q'(v)),\ \{P\}\ e\ \{\lambda w.Q(w)\} \vdash \{P'\}\ e\ \{\lambda w.Q'(w)\}$ (consequence)
- $\{P\}\ e\ \{\lambda v.Q(v)\},\ (\forall v.\{Q(v)\}\ K[v]\ \{\lambda w.R(w)\}) \vdash \{P\}\ K[e]\ \{\lambda w.R(w)\}$ (bind)
- $\forall x.\{P(x)\}\ e\ \{\lambda v.Q(v)\} \vdash \{\exists x.\ P(x)\}\ e\ \{\lambda v.Q(v)\}$ (exists)

예시들


$\{n < 2^{64} - 1\}$
$\{\text{True}\}$
consequence
bind + spec-min
$\textbf{let } m := \min\ 0\ \overline{n} \textbf{ in}$
$\{\exists p_m.\ m = \overline{p_m} \wedge p_m \leq 0 \wedge p_m \leq n\}$
consequence
$\{\overline{m} + 1 < 0\}$
$\textbf{let } y := \text{add}\ m\ 1 \textbf{ in}$
bind + spec-add
$\{y = \overline{m+1}\}$
y
value
$\{y = \overline{p_m + 1} \wedge p_m + 1 \leq 1\}$
consequence
$\{\exists (p : \mathbb{Z}).\ y = \overline{p} \wedge p \leq 1)\}$
$\text{and} = \lambda b_1, b_2.\ \textbf{if } b_1 \textbf{ then } b_2 \textbf{ else } \text{false}$
$\text{add} = \lambda x, y.\ x + y$
$\min = \lambda x, y.\ \textbf{if } x < y \textbf{ then } x \textbf{ else } y$
$\{\text{True}\}\ \text{and}\ b_1\ b_2\ \{\lambda v.\ v = \overline{b_1\ \&\ b_2}\}$
$\{n + m < 2^{64}\}\ \text{add}\ \overline{n}\ \overline{m}\ \{\lambda v.\ v = \overline{n+m}\}$
$\{\text{True}\}\ \min\ \overline{n}\ \overline{m}\ \{\lambda v.\ \exists (p : \mathbb{Z}).\ v = \overline{p} \wedge p \leq n \wedge p \leq m\}$

명령형 언어에 대한 논증?

$$
\begin{aligned}
\text{Values} \quad v &::= \lambda x.\, e \mid \overline{n} \mid \text{true} \mid \text{false} \mid (v_1, v_2) \\
&\mid () \mid \ell \\
\\
\text{Expressions} \quad e &::= x \mid v \mid \lambda x.\, e \mid e_1\, e_2 \\
&\mid \textbf{if}\, e\, \textbf{then}\, e_1\, \textbf{else}\, e_2 \\
&\mid e_1 + e_2 \mid e_1 == e_2 \mid e_1 < e_2 \\
&\mid (e_1, e_2) \mid \pi_1\, e \mid \pi_2\, e \\
&\mid \text{alloc}\ e_1 \mid !\, e_1 \mid e_1 \leftarrow e_2
\end{aligned}
$$

명령형 언어에 대한 논증?

- 새로운 연결사 도입: l ↦ v
  - 의미: l이라는 위치값에 v라는 값이 저장되어 있음
- 직관
  - {x ↦ v ∧ y ↦ w} y ← 42 {x ↦ v ∧ y ↦ 42}
  - 의미: y라는 위치에 정수 42를 저장
- 만약 두 위치값 x와 y가 사실 같은 값이라면
  - x에 대한 전건이 그대로 후건에서 보존될 수 없음
- 해결책?
  - {x ↦ v ∧ y ↦ w ∧ x ≠ y} y ← 42 {x ↦ v ∧ y ↦ 42}
  - 할당된 위치값이 100개라면?

$$
\begin{array}{lll}
\text{Values} & v ::= \lambda x.\, e \mid \overline{n} \mid \text{true} \mid \text{false} \mid (v_1, v_2) \\
& \mid () \mid \ell \\
\\
\text{Expressions} & e ::= x \mid v \mid \lambda x.\, e \mid e_1\, e_2 \\
& \quad \mid \textbf{if}\ e\ \textbf{then}\ e_1\ \textbf{else}\ e_2 \\
& \quad \mid e_1 + e_2 \mid e_1 == e_2 \mid e_1 < e_2 \\
& \quad \mid (e_1, e_2) \mid \pi_1\, e \mid \pi_2\, e \\
& \quad \mid \text{alloc}\ e_1 \mid !\, e_1 \mid e_1 \leftarrow e_2
\end{array}
$$

분리논리의 도입

$$\text{Propositions} \qquad P ::= \lceil \phi \rceil \mid \exists x.\, P(x) \mid \forall x.\, Q(x) \mid P \vee Q$$
$$\mid \ell \mapsto v \mid P \star Q \mid \text{emp}$$

- l ↦ v
  - l ‘points-to’ v
  - 위치 l이 값 v를 저장하고 있음
- P ★ Q
  - Separating conjunction, P ‘star’ Q
  - 메모리의 겹치지 않는 부분이(disjoint) 각각 P와 Q를 만족함
- emp
  - 메모리가 비어 있음

규칙들

$$P \star Q \vdash Q \star P \qquad \text{sep-comm}$$

$$P \star (Q \star R) \vdash (P \star Q) \star R \qquad \text{sep-assoc}$$

$$(\exists x.\, P(x)) \star Q \vdash (\exists x.\, P(x) \star Q) \qquad \text{sep-exists}$$

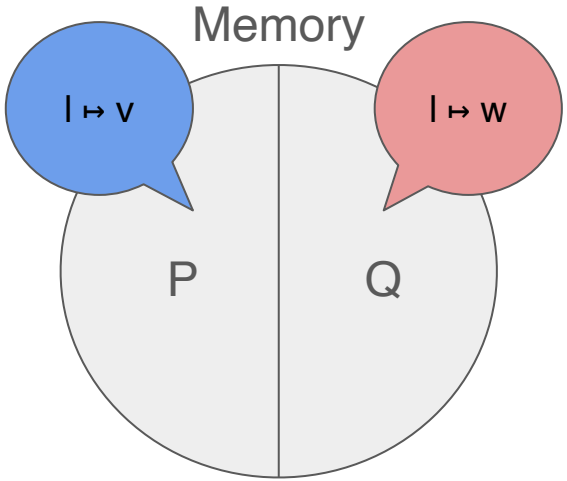
$$\ell \mapsto v \star \ell \mapsto w \vdash \text{False} \qquad \text{pointsto-sep}$$

$$P \vdash P \star \text{emp} \qquad \text{sep-id}$$

$$\frac{\lceil \phi \rceil}{P \vdash P \star \lceil \phi \rceil} \quad \text{sep-pure}$$

$$P \star \lceil \phi \rceil \vdash P \quad \text{sep-pure-weaken}$$

$$\frac{Q \vdash Q'}{P \star Q \vdash P \star Q'} \quad \text{sep-monotone}$$



호어 논리의 문제 해결

- $\{x \mapsto v \land y \mapsto w\}\ y \leftarrow 42\ \{x \mapsto v \land y \mapsto 42\}$
  - 호어 논리에서 위와 같은 호어 삼중항은 성립하지 않음
  - x와 y라는 변수가 사실은 같은 위치값을 가지고 있는 경우를 배제할 수 없기 때문에
  - 따라서 포인터가 있는 언어를 쉽게 검증할 수 없음
- $\{x \mapsto v \star y \mapsto w\}\ y \leftarrow 42\ \{x \mapsto v \star y \mapsto 42\}$
  - 메모리의 '겹치지 않는 부분이 각각' $x \mapsto v$와 $y \mapsto w$를 저장 중
  - 전건으로부터 x와 y는 서로 다른 위치값이라는 사실을 도출할 수 있음
  - 따라서 후건에서 x가 저장하고 있는 값은 변하지 않음

$$\{\mathrm{emp}\}\ \mathrm{alloc}\ v\ \{\lambda w.\, \exists \ell.\, \lceil w = \ell \rceil \star \ell \mapsto v\}\quad \text{alloc-spec}$$

$$\{\ell \mapsto v\}\ !\ell\ \{\lambda w.\, \lceil w = v \rceil \star \ell \mapsto v\}\quad \text{load-spec}$$

$$\{\ell \mapsto v_0\}\ \ell \leftarrow v\ \{\lambda \_.\, \ell \mapsto v\}\quad \text{store-spec}$$

프레임 규칙

Frame problem

9 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

In artificial intelligence, with implications for cognitive science, the **frame problem** describes an issue with using first-order logic to express facts about a robot in the world. Representing the state of a robot with traditional first-order logic requires the use of many axioms that simply imply that things in the environment do not change arbitrarily. For example, Hayes describes a "block world" with rules about stacking blocks together. In a first-order logic system, additional axioms are required to make inferences about the environment (for example, that a block cannot change position unless it is physically moved). The frame problem is the problem of finding adequate collections of axioms for a viable description of a robot environment.[1]

- f는 l의 소유권을 요구, f의 실행 내내 $l \mapsto 0$를 소유함
- 반대로 l'의 소유권을 요구하지 않으므로, l'에 접근하지 않는다는 것도 알 수 있음

예시

$$\{\ell \mapsto \overline{0}\}\quad f\,(\ell, \ell')\quad \{\lambda\_.\, \ell \mapsto \overline{42}\}$$

$$\begin{array}{l} e_{\text{own}} ::= \\ \quad \textbf{let}\ x := \text{alloc}\ \overline{0}\ \textbf{in} \\ \quad \textbf{let}\ y := \text{alloc}\ \overline{42}\ \textbf{in} \\ \quad f\,(x, y); \\ \quad \text{assert}\,(!\,x == !\,y) \end{array}$$

$$\{\text{True}\}\ e_{\text{own}}\ \{\lambda\_.\, \text{True}\}$$

$$\begin{array}{l} \{\text{True}\} \\ \quad \textbf{let}\ x := \text{alloc}\ \overline{0}\ \textbf{in} \\ \{x \mapsto \overline{0}\} \\ \quad \textbf{let}\ y := \text{alloc}\ \overline{42}\ \textbf{in} \\ \{x \mapsto \overline{0} \star y \mapsto \overline{42}\} \\ \quad f\,(x, y) \\ \{x \mapsto \overline{42} \star y \mapsto \overline{42}\} \\ \quad \text{assert}\,(!\,x == !\,y) \rightsquigarrow \\ \quad \text{assert}\,(!\,x == \overline{42}) \rightsquigarrow \\ \quad \text{assert}\,(\overline{42} == \overline{42}) \\ \{x \mapsto \overline{42} \star y \mapsto \overline{42}\} \\ \quad () \\ \{\text{True}\} \end{array}$$

Iris project(2015)

- Rocq 증명보조기에서 형식화된 분리논리체계
  - 믿을 수 있는 증명
- 프로그램의 여러 성질
  - 안전성(Safety), 결국성질(Liveness property), …
- 동시성 프로그램
  - 분리논리의 장점: 여러 실행이 동시에 발생하는 프로그램의 검증에 알맞음
- 고차 함수 프로그램
- 분산시스템, 운영체제, Rust 언어의 타입 안전성, …
- 10년간 140편 이상의 논문 발표

예언자 변수(Prophecy variables)

- 프로그램의 미래 행동을 알아야 제대로 논증할 수 있는 경우 발생
  - 미래 행동을 기록하는 변수
- 분리논리에 도입
  - The Future is Ours: Prophecy Variables in Separation Logic (Jung et al. 2020)
- 문제
  - 예언자 변수는 호어 삼중항으로 주어진 프로그램의 미래 행동을 기술할 수 없음
  - 한정된 형태로만 사용될 수 있음
- CCR (Song et al., 2023) 프레임워크를 활용하여 해결

인공지능과의 접점

- 프로그램 논리를 사용한 소프트웨어 엄밀 검증의 최대 약점
  - 힘들고 어렵고 시간이 오래 걸리고 비쌈
  - 반복문 불변량(Loop invariant)
  - 많은 인력이 투입되어야 함
- AI + Proof assistant
  - 믿을 수 없는 인공지능과 믿을 수 있는 증명보조기
  - 믿을 수 없는 소프트웨어에 대한 믿을 만한 증명 => 믿을 만한 소프트웨어
- Wu et al. ICLR 2024, …

참고자료

https://plv.mpi-sws.org/semantics-course/lecturenotes.pdf

https://tchajed.github.io/sys-verif-fa25/

감사합니다.