Subject: تحلیل برنامه (درس دوم)
Year: ۱۳۹۹ Month: ۱۱ Date: ۲۱

Program Analysis :

Formal Verification of Programs : عنوان دوره

Textbooks:

1- "Principles of Program Analysis", Nilsen, 2005, 2nd Ed. Springer
(PPA)

MIT ← کتاب درسی که برای تدریس این درس استفاده می‌شود

Subject Venue:
(PLDI, POPL, TOPLAS مجلات)

تحلیل برنامه عمدتا از نوع Statically بوده و بخش عمده کاربرد دارد

تحلیل‌های برنامه را با Testing از چه جهاتی مقایسه می‌کنید؟

تحلیل:
- Static → خودکار، کامل‌تر
- Dynamic → برای همه برنامه‌ها قابل اجرا نیست، اجرا می‌شود
- Run-time → برنامه‌ها را اجرا می‌کند، اجرا می‌شود

Automate بودن تحلیل‌ها (خودکار) ↓ Scalable بودن

(Runtime Verification)

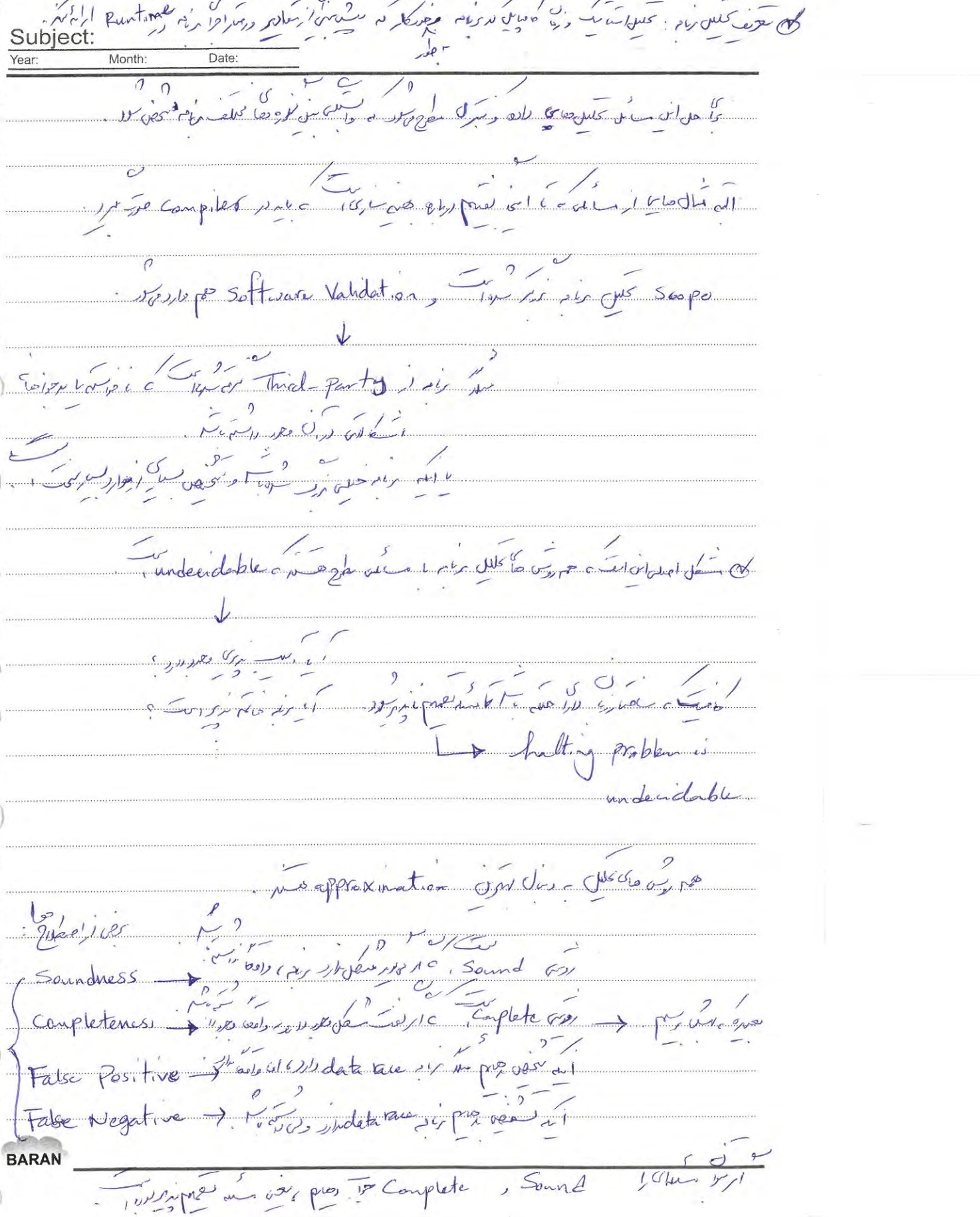
Superfluous computation, Redundancy از چه چیزهایی جلوگیری می‌کند؟

redundancy ← { محاسبه مجدد نتایج محاسبه شده / اجرای مجدد loop invariant ها }

Dead Code BARAN

Subject:

Year: Month: Date:



BARAN

Subject: کلیات زبان (The infinite ...)

Year: 1397 Month: 11 Date: 21

Completeness و Soundness تعریف دقیق این دو ویژگی در تعریف یک تحلیل ...

در عمل معمولاً Soundness را فدای Completeness می‌کنند (یعنی اجازه می‌دهند FN داشته باشد ولی FP نداشته باشد).

Practical

در روش‌های عملی ← کمی از Soundness و کمی از Completeness فدا می‌کنند.

Completeness, Soundness را می‌توان به شکل دقیق‌تری "more precise" تعریف کرد.

اگر Soundness را حذف کنیم روش useless می‌شود. FP زیاد روش را

Completeness را ثابت نگه می‌داریم.

(مثال: Pixy و Satura از ابزارهای تحلیل ... هستند)

در عمل معمولاً به دنبال approximation هستیم که در آن FP را تا حد امکان کم کنیم.

در این درس بیشتر روی Soundness تمرکز می‌کنیم.

تحلیل‌های Probabilistic ← احتمال وقوع یک مسیر اجرایی

Appendix ... ppa ... را مطالعه کنید.

Subject: Grading: Midterm 30% Assignments 10%
Year: Month: Date: Final 50% project 10%

Other Textbooks:

2 - Winskel, Foundations of Programming Languages

mathematical Preliminaries

3 - Pierce, Types and Programming Languages

4 - Baier, Katoen, Principles of Model Checking

2008

5 - Bradley, Mana, The Calculus of Computation

2007

Formal Verification

Temporal Logic

6 - David Kroening, Decision Procedures, Springer, 2008.

Introduction:

developing a feel!

Subject: تحلیل برنامه

Year: ۹۴ Month: ۱۱ Date: ۲۱

Example —

(زیرا که این است که y مقدار نمی‌گیرد.)

```
read(x);
(if x>0 then y:=1 else (y:=2; S));
z:=y;
```

↳ چه مقادیری از y می‌تواند به این قسمت برسد.

(کار تحلیل یعنی همین پیش‌بینی)

جواب ساده: با نگاه به برنامه می‌بینیم که مقدار y می‌تواند 1 یا 2 باشد.

اگر بعضی پس از تحلیل بگویند مقدار متغیر 1 یا 2 است، آیا حرف غلطی است؟

↓

اگر S را فرض کنیم، $x \leq 0$ و $y = 2$ بوده،

non-terminating باشد!

پس اگر واقعاً S هیچ موقع تمام نشود، حالا جواب دقیق این است که 1! کجا مقدار 6 درست

پس اگر بگوییم y می‌تواند 1 و 2 باشد، دوباره پاسخ درستی می‌دهیم. حتی اگر بگوییم 1، 2 و 7 هم باز درست است.

این هم پاسخ درستی است، اما به دقت آن نیست. پس کلی‌ترین حالت این است که دقیق‌ترین جواب ممکن به دست آید.

می‌دانیم که تحلیل همه چیز را نمی‌داند، پس فقط می‌خواهیم پاسخ درستی بدهیم.

Subject:

Year: Month: Date:

The values of y possible at 2:=y are among 1 and 2.

While Language :

elementary blocks:

assignment, Conditional classes, and Skip (test)

abstract syntax

(label)

$a \in AExp$ (arithmatic expressions)

$b \in BExp$ (binary expressions)

$S \in Stmt$ (statements)

$x, y \in Var$ (Variables)

$n \in Num$ (Numerals)

$\ell \in Lab$ (Labels)

$op_a \in Op_a$ (arithmetic operations)

$op_b \in Op_b$ (binary operations)

$op_r \in Op_r$ (relational operations)

Subject: ریز جلسه

Year: 99 Month: 11 Date: 21

$a ::= x \mid n \mid a_1 \; op_a \; a_2$

$b ::= true \mid false \mid not\ b \mid b_1 \; op_b \; b_2 \mid a_1 \; op_r \; a_2$

$S ::= [x := a]^l \mid [skip]^l \mid S_1 ; S_2 \mid if\ [b]^l\ then\ S_1\ else\ S_2$

$\mid while\ [b]^l\ do\ S$

هر برچسب (label) متناظر با یک block در برنامه است.

Example -

$[y := x]^1 ; [z := 1]^2 ; while\ [y > 1]^3\ do\ ([z := z * y]^4 ;$

$[y := y - 1]^5) ; [y := 0]^6$

Reaching Definition Analysis (تحلیل تعاریف دست‌یابنده)

↓

assignment (Def - Use)

تحلیلی برای یافتن مکان‌هایی که یک متغیر در آن مقداردهی شده است.

An assignment (definition) of the form $[x := a]^l$ may reach a certain program point if there is an execution of the program where x was last assigned a value at l when the program point is reached.

BARAN

Subject: ٤,٢ →

Year: Month: Date:

(Canonical Analysis)

Data Flow Analysis ... reaching definitions ...

reaching definitions ...

l	$RD_{entry}(l)$
1	$(x,?),(y,?),(z,?)$
2	$(x,?),(y,1),(z,?)$
3	$(x,?),(y,1),(y,5),(z,2),(z,4)$
4	
5	
6	

uninitialized

l	$RD_{exit}(l)$
1	$(x,?),(y,1),(z,?)$
2	$(x,?),(y,1),(z,2)$
3	$(x,?),(y,1),(y,5),(z,2),(z,4)$
4	
5	
6	

... $y:=1$ ...

... $y:=y*1$ ...

... data flow ... RD ...

... equational ... (معادلات)

$$RD = (RD_{entry}, RD_{exit})$$

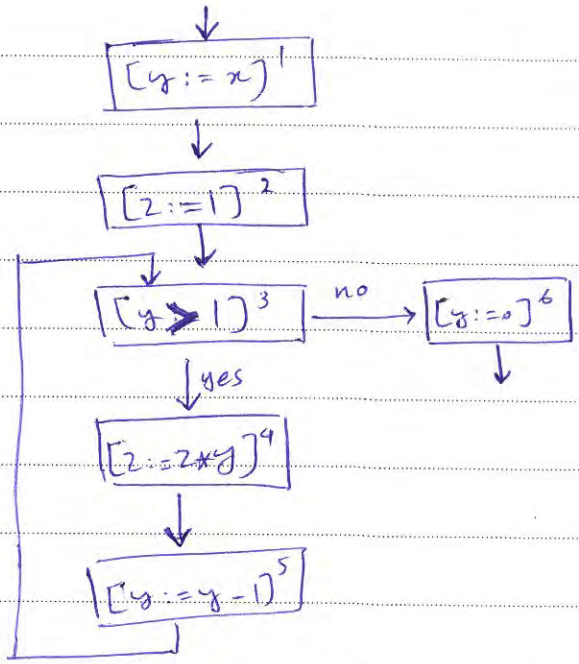
$$RD_{entry}(3) = \{(x,?),(y,1),(y,5),(z,2),(z,4)\}$$

$$RD_{exit}(3) = RD_{entry}(3)$$

BARAN

Subject: PA
Year: 1394 Month: 11 Date: ۲۲

Control flow graph



۱) تعاریف Reaching Def. حذف و تولید مجموعه‌های RD را به صورت زیر تعریف می‌کنیم:

$$RD_{exit}(1) = (RD_{entry}(1) \setminus \{(y,?) \mid ? \in Lab\}) \cup \{(y,1)\}$$

$$RD_{exit}(2) = (RD_{entry}(2) \setminus \{(z,?) \mid ? \in Lab\}) \cup \{(z,2)\}$$

$$RD_{exit}(3) = RD_{entry}(3)$$

$$RD_{exit}(4) = (RD_{entry}(4) \setminus \{(z,?) \mid ? \in Lab\}) \cup \{(z,4)\}$$

BARAN

$RD_{exit}(5) = \ldots$ $RD_{exit}(6) = \ldots$

Subject:

Year: Month: Date:

۲) معادلات RD_{entry} بر حسب RD_{exit} برای هر بلوک را بنویسید.

$$RD_{entry}(1) = \{(x,?),(y,?),(z,?)\}$$

$$RD_{entry}(2) = RD_{exit}(1)$$

$$RD_{entry}(3) = RD_{exit}(2) \cup RD_{exit}(5)$$

$$RD_{entry}(4) = RD_{exit}(3)$$

$$RD_{entry}(5) = RD_{exit}(4)$$

$$RD_{entry}(6) = RD_{exit}(3)$$

هر نوع معادله بازگشتی که حل شود و فرم بسته آن را بدست آورد.

$RD_{entry}(1), \ldots, RD_{exit}(6)$ ← مجهول

این مجهول‌ها را می‌توان به صورت بردار $\overrightarrow{RD}$ نشان داد.

سپس می‌توان معادله را به صورت زیر نوشت:

$$\overrightarrow{RD} = F(\overrightarrow{RD})$$

تابعی است بر حسب بردار مجهول که می‌خواهیم نقطه ثابت آن را بیابیم.

BARAN

Subject: PA

Year: 1394 Month: 11 Date: ۲۴

$$F(\overrightarrow{RD}) = \left(F^{(1)}_{entry}(\overrightarrow{RD}), \ldots, F^{(6)}_{exit}(\overrightarrow{RD}) \right)$$

e.g. $F^{(3)}_{entry}(\ldots, RD_{ent}(2), \ldots, RD_{exit}(5), \ldots) = RD_{ent}(2) \cup RD_{exit}(5)$

... F ... fixed Point ...

$$F : \left(\wp(Var_* \times Lab_*) \right)^{12} \longrightarrow \left(\wp(Var_* \times Lab_*) \right)^{12}$$

... (ref, antisym, trans...)

... partial order set ... $\sqsubseteq$ ... $\left(\wp(Var_* \times Lab_*) \right)^{12}$ ...

$$\overrightarrow{RD} \sqsubseteq \overrightarrow{RD}' \text{ iff } RD_i \subseteq RD'_i$$

$\overrightarrow{RD}$ ...

... Complete lattice ...

... lub ... glb ...

... (monoton) ...

... Complete lattice ...

... lub ... glb ...

... (Monoton Framework) ...

BARAN

Subject:

Year: Month: Date:

$\vec{RD} \sqcup \vec{RD'} = (RD_1 \sqcup RD'_1, \ldots, RD_{12} \sqcup RD'_{12})$ (اعضا)

lub

$\vec{\phi} = (\phi, \phi, \ldots, \phi)$ ← کوچکترین glb

(12 تا)

تابع F، تغیر (monotone) نیست، یعنی:

$\vec{RD} \sqsubseteq \vec{RD'}$ implies $F(\vec{RD}) \sqsubseteq F(\vec{RD'})$

(برای F صعودی)

بعد از F (3) entry

این یعنی 12 تا تابع صعودی نمی‌تونن داده‌ساختار بسازن.

دنباله $(F^n(\vec{\phi}))$ را در نظر بگیرید.

$(\vec{\phi}, F(\vec{\phi}), F(F(\vec{\phi})), \ldots)$ ← اعداد دنباله

n ↓

مجموعه $\wp(Var \times Lab)^{12}$ تشکیل یک دامنه کامل می‌دهد.

تساوی‌پذیری ندارد و چون F صعودی است MAX را برمی‌گرداند.

$\forall n.\ F^n(\vec{\phi}) \sqsubseteq F^{n+1}(\vec{\phi})$ (اثبات نشده) صعودی

$\exists n.\ F^{n+1}(\vec{\phi}) = F^n(\vec{\phi})$ پس یک $\vec{RD}$ وجود دارد که MAX و نقطه ثابت است.

چون نقطه ثابت است

$F^{n+1}(\vec{\phi}) = F(F^n(\vec{\phi}))$

$\vec{RD}$ نقطه ثابت است

BARAN

جواب تمرین‌ها در مورد Solution

Subject: PA

Year: ۱۳۹۲ Month: ۱۱ Date: ۲۲

$F^{\uparrow}(\vec{\phi})$ is the least solution of $\vec{RD} = F(\vec{RD})$.

↳ (Safe approximation : برای تضمین درستی approximation)

$$F^{n}(\vec{\phi}) \sqsubseteq \vec{RD}$$

↓

solution

$$\vec{\phi} \sqsubseteq \vec{RD}$$

$$F(\vec{\phi}) \sqsubseteq F(\vec{RD}) = \vec{RD}$$

$$\vdots$$

$$F^{\uparrow}(\vec{\phi}) \sqsubseteq \vec{RD}$$

این ها را برای معادلات می‌توان از روی Constraint ها نوشت.

$$\vec{RD} \sqsupseteq F(\vec{RD})$$

↓

می‌توان یکی از جواب‌های بزرگتر را هم حساب کرد.

↓

Super set (مجموعه‌ای)

$\vec{RD} \sqsupseteq F(\vec{RD})$ یعنی

این یک تضمین است ولی ممکن است خیلی بزرگ باشد.

BARAN

۱۴

Subject:
Year: ۲۰۲۲ Month: ۱۱ Date: ۲۸

The least solution

The same least solution $F^n(\vec{\emptyset})$

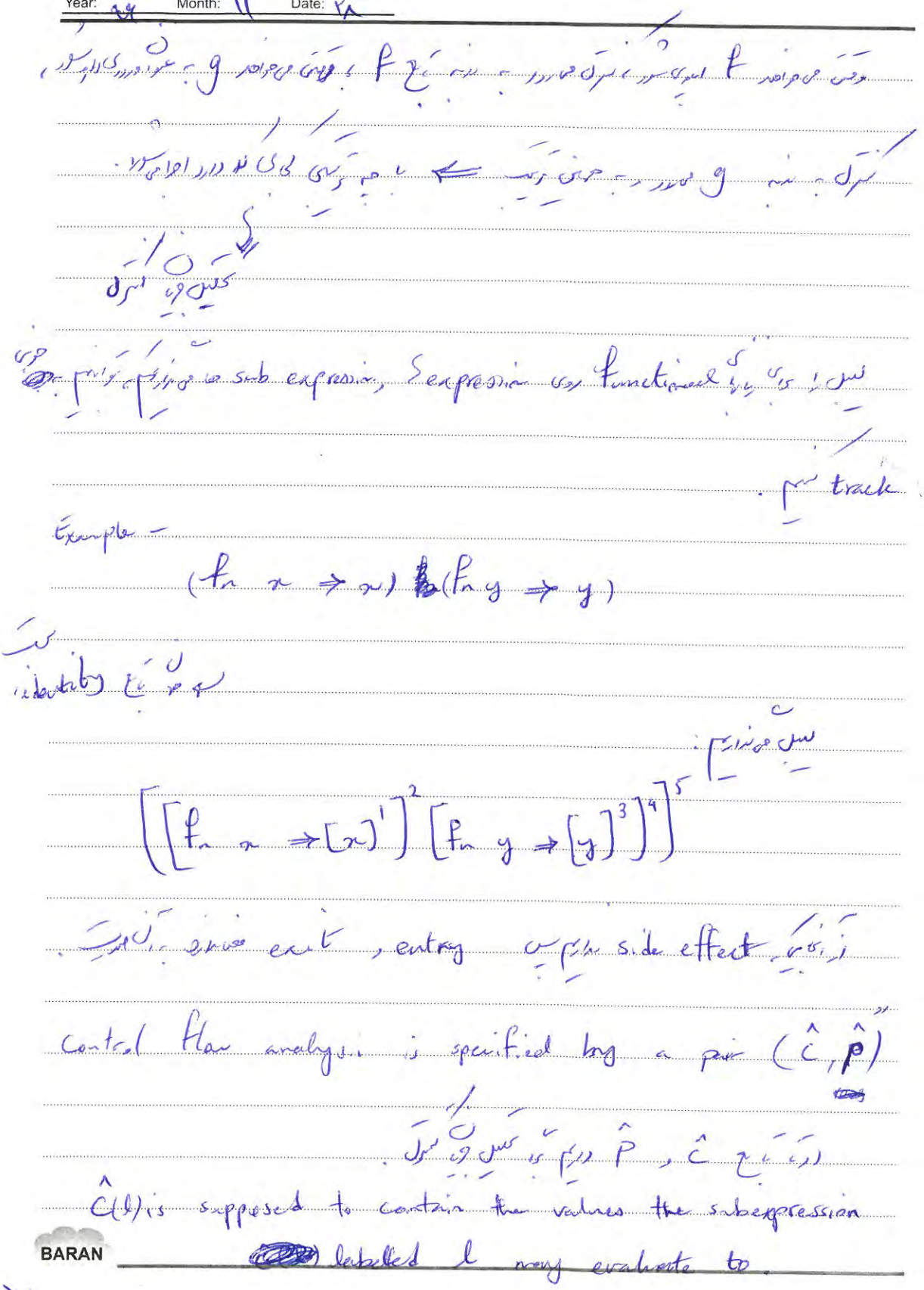
Constraint-Based Analysis

Control Flow Analysis

Typical

Subject: PA

Year: ۲۰۲۴ Month: 11 Date: ۲۸



Subject:

Year: Month: Date:

$\hat{c}(l)$ مشخص می‌کند که متغیر subexp در برچسب l به چه چیزهایی می‌تواند bound شود.

$\hat{\rho}(x)$ contains the values that the variable x can be bound to.

مقادیری که متغیر x می‌تواند bound شود.

۳ زیر مجموعه داریم:

۱) هر abstraction که در برنامه وجود دارد و متغیر subexp آن را bound می‌کند:

$\{fn\ x \Rightarrow [x]^1\} \subseteq \hat{c}(2)$

$\{fn\ y \Rightarrow [y]^3\} \subseteq \hat{c}(4)$

چون خود abstraction ها هم value هستند.

۲) تمام متغیرهایی که در برنامه bound شده‌اند و متغیرهایی که در برنامه آزاد هستند:

$\hat{\rho}(x) \subseteq \hat{c}(1)$ آن را بررسی کند.

$\hat{\rho}(y) \subseteq \hat{c}(3)$ چون خود متغیر است، مقدار دارد.

۳) برای application ها $[e_1\ e_2]^l$ که در برنامه وجود دارند:

Subject: PA

Year: 94 Month: 11 Date: ۲۸

I. a constraint expressing that the formal parameter of the function is bound to the actual parameter at the application point.

II. a constraint expressing that the result obtained by evaluating the body of the function is a possible result of the application.

$f_n\ x \Rightarrow (x)^1 \qquad f_n\ y \Rightarrow (y)^3$

$\hat{c}(2) \subseteq \{f_n\ x \Rightarrow (x)^1,\ f_n\ y \Rightarrow (y)^3\}$

$\hat{c}(4) \subseteq \hat{\rho}(x)$

$\hat{c}(1) \subseteq \hat{c}(5)$

$\{f_n\ x \Rightarrow (x)^1\} \subseteq \hat{c}(2) \Rightarrow \hat{c}(4) \subseteq \hat{\rho}(x)$

$\{f_n\ x \Rightarrow (x)^1\} \subseteq \hat{c}(2) \Rightarrow \hat{c}(1) \subseteq \hat{c}(5)$

BARAN

۱۷/

Subject:

Year: Month: Date:

$\{f_n\ y \Rightarrow (y)^3\} \subseteq \hat{c}(2) \Rightarrow \hat{c}(4) \subseteq \hat{\rho}(y)$

$\{f_n\ y \Rightarrow [y]^3\} \subseteq \hat{c}(2) \Rightarrow \hat{c}(3) \subseteq \hat{c}(5)$

سپس آنالیز ما شامل این قوانین می‌شود:

$\hat{c}(1) = \{f_n\ y \Rightarrow [y]^3\}$

$\hat{c}(2) = \{f_n\ x \Rightarrow [x]^1\}$

$\hat{c}(3) = \phi$

$\hat{c}(4) = \{f_n\ y \Rightarrow [y]^3\}$

$\hat{c}(5) = \{f_n\ y \Rightarrow (y)^3\}$

$\hat{\rho}(x) = \{f_n\ y \Rightarrow [y]^3\}$

$\hat{\rho}(y) = \phi$

برای SMT Solver یا Constraint Solver یا theory Solver

$\hat{\rho}(y) = \phi$ یعنی apply شده $f_n\ y \Rightarrow y$ در برنامه.

(یعنی bound برای y)

BARAN

Subject: PA

Year: ۹۹ Month: ۱۱ Date: ۲۸

The program may evaluate to $\hat{C}(5)$ only.

Semantics-based ⟸ Safe ⟸ کلیه روش‌های تحلیل ایستا

↓

روی برنامه‌ها اجرا نمی‌شود

Abstract Interpretation

(تفسیر انتزاعی، تفسیر مجرد)

تحلیل ایستا با استفاده از روش‌های رسمی مبتنی بر Semantics

(Semantics مبتنی بر معنای برنامه meaning)

Semantics استاندارد

(Abstract Interpret) non-standard Semantics

↓

تحلیل برنامه بدون اجرای برنامه، معمولا بر مبنای ترکیبی از ساختار برنامه

↓

تحلیل جریان داده Data Flow

تحلیل جریان کنترل Control Flow

BARAN

۹/

Subject:
Year: Month: Date:

Abstract Noninterference → [illegible] Collecting semantics ←

[illegible]

[illegible] specify [illegible]

[illegible]

Collecting Semantics:

It records the set of traces "tr" that can reach a given program point.

$$tr \in Trace = (Var \times Lab)^*$$

[illegible] Data Flow [illegible]

[illegible] trace [illegible]

[illegible] trace [illegible]

$[y := x]^1; [z := 1]^2; \text{while } [y > 1]^3 \text{ do } ([z := z * y]^4; [y := y - 1]^5); [y := 0]^6$

[illegible] trace [illegible]:

$((x,?),(y,!),(z,?),(y,1),(z,2),(z,4),(y,5),(y,6))$

BARAN

Subject: PA

Year: 99 Month: 11 Date: ۲۸

mapping

trace

Semantic Reaching Definitions

$SRD(tr)(x) = l$ iff the right most pair (x, l') in tr has $l = l'$.

$DOM(tr) = \{x \mid SRD(tr)(x) \text{ is defined}\}$

$$\forall x \in DOM(tr) \cdot (x, SRD(tr)(x)) \in RD_{entry}(l)$$

tr is the trace just before entering the elementary block labelled l.

Subject: Hyper Collecting / Calculational Design of CSP Runtime Monitor 2017

Year: Month: Date: Sept. 18

Galois Connection

$$\left(\wp(Trace)\right)^{12}$$

$$CS_{exit}(1) = \{tr : (y,1) \mid tr \in CS_{entry}(1)\}$$

$$CS_{exit}(2) = \{tr : (2,2) \mid tr \in CS_{entry}(2)\}$$

$$CS_{exit}(3) = CS_{entry}(3)$$

$$CS_{exit}(6) = \{tr : (y,6) \mid tr \in CS_{entry}(6)\}$$

$$CS_{entry}(1) = \{((x,?),(y,?),(z,?))\}$$

$$CS_{entry}(2) = CS_{exit}(1)$$

$$CS_{entry}(3) = CS_{exit}(2) \cup CS_{exit}(5)$$

$$CS_{entry}(6) = CS_{exit}(3)$$

BARAN

Subject: PA

Year: ۲۰۲۴ Month: ۱۱ Date: ۳۰

$$\overrightarrow{CS} = G(\overrightarrow{CS})$$

$$G:\left(\wp(Traces)^{12}\right) \rightarrow \left(\wp(Traces)\right)^{12}$$

Subject:
Year: Month: Date:

reaching definition

Galois Connection:

abstract

Lab, Var

$$\wp(\text{Trace}) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \wp(\text{Var} \times \text{Lab})$$

(concrete) (abstract)

Reaching Definition

α : abstraction function

RD

γ : Concretization function

trace

a set of traces

$$\alpha(X) = \{ (x, SRD(tr)(x)) \mid x \in DOM(tr) \wedge tr \in X \}$$

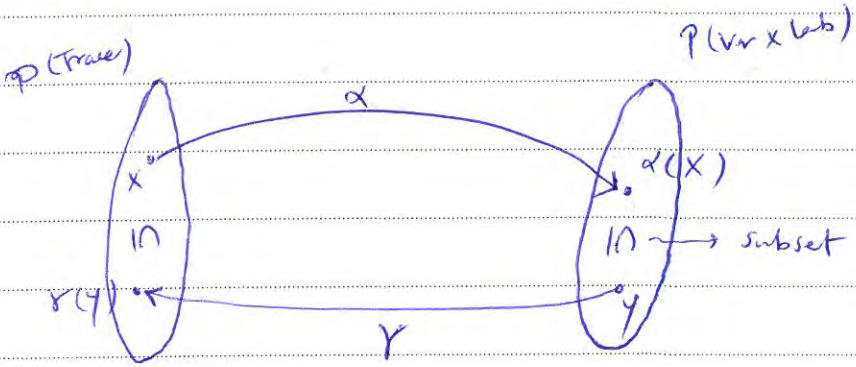
$$\gamma(Y) = \{ tr \mid \forall x \in Dom(tr). (x, SRD(tr)(x)) \in Y \}$$

BARAN

Subject: PA
Year: 94 Month: 11 Date: ۲۰

often it is demanded that α and γ satisfy

$$\alpha(X) \subseteq Y \Leftrightarrow X \subseteq \gamma(Y)$$



(α, γ) is called an adjuction or Galois Connection

if the above condition is satisfied.

Induced Analysis:

$\alpha, \gamma, G_1 \rightsquigarrow$ Reaching Definition

Subject:

Year: Month: Date:

$$\vec{\alpha}(X_1, X_2, \ldots, X_{12}) = (\alpha(X_1), \ldots, \alpha(X_{12}))$$

$$\vec{\gamma}(y_1, \ldots, y_{12}) = (\gamma(y_1), \ldots, \gamma(y_{12}))$$

$$(\vec{\alpha} \circ G \circ \vec{\gamma}) : (\mathcal{P}(\text{Var} \times \text{Lab}))^{12} \to (\mathcal{P}(\text{Var} \times \text{Lab}))^{12}$$

(...) Reaching Definitions ...

... fixed point ...

...

$$CS_{exit}(4) = \{ tr : (2,4) \mid tr \in CS_{entry}(4) \}$$

$$G_{exit}(4)(\ldots, CS_{entry}(4), \ldots) = \{ tr : (2,4) \mid tr \in CS_{entry}(4) \}$$

$$\alpha(G_{exit}(4)(\vec{\gamma}(\ldots, RD_{entry}(4), \ldots))) = \alpha(\{ tr : (2,4) \mid tr \in \gamma(RD_{entry}(4)) \})$$

$$= \{ (x, SRD(tr:(2,4))(x)) \mid x \in Dom(tr:(2,4)) \wedge \forall y \in Dom(tr).\ (y, SRD(tr)(y)) \in RD_{entry}(4) \}$$

$$= \{ (x, SRD(tr:(2,4))(x)) \mid x \neq z \wedge x \in Dom(tr:(2,4)) \wedge \forall y \in Dom(tr).\ (y, SRD(tr)(y)) \in RD_{entry}(4) \}$$

$\alpha(X)$ ...

(4)

Subject: PA

Year: 99 Month: 11 Date: ۲۰

$$\bigcup \{(x, SRD(tr:(2,4))(x)) \mid x = 2 \wedge x \in Dom(tr:(2,4)) \wedge \forall y \in Dom(tr) . (y, SRD(tr)(y)) \in RD_{entry}(4)\}$$

$$= \{(x, SRD(tr))(x) \mid x \in Dom(tr) \wedge x \neq 2 \wedge \forall y \in Dom(tr) . (y, SRD(tr)(y)) \in RD_{entry}(4)\}$$

$$\bigcup \{(2,4) \mid \forall y \in Dom(tr) . (y, SRD(tr)(y)) \in RD_{entry}(4)\}$$

$$= RD_{entry}(4) \setminus \{(2,\ell) \mid \ell \in Lab\} \cup (2,4)$$

$$RD_{exit}(4) = RD_{entry}(4) \setminus \{(2,\ell) \mid \ell \in Lab\} \cup (2,4)$$

$$\overrightarrow{RD} = (\vec{\alpha} \circ c \circ \vec{\gamma})(\overrightarrow{RD})$$

$$lfp(\vec{\alpha} \circ c \circ \vec{\gamma})$$

BARAN

۲۷

Subject:

Year: Month: Date:

precedes

$(\vec{\alpha} \circ G \circ \vec{\gamma}) \sqsubseteq F$

$\vec{\alpha}(G\hat{}(\vec{\phi})) \sqsubseteq (\vec{\alpha} \circ G \circ \vec{\gamma})\hat{}(\vec{\phi}) \sqsubseteq F\hat{}(\vec{\phi})$

پس:

$\vec{\alpha}(lfp(G_1)) \sqsubseteq lfp(\vec{\alpha} \circ G \circ \vec{\gamma}) \sqsubseteq lfp(F)$

تحلیل انتزاعی — تحلیل RD

$lfp(F)$ ... collecting Semantics (G_Γ) ...

... G_Γ ...

... $\vec{\alpha}(lfp(G_\Gamma))$ ...

... (RD ...!)

... FP ...

N. Isen

... Type and Effect ...

Subject: PA
Year: ۱۳۹۷ Month: ۱۲ Date: ۵

Type and Effect Systems:

syntactic ... tractable ... type ...

light-weight ... type ...

functional ... discipline ... type ...

effect ... evaluation ... assignment ...
(side effect)
I/O channels

type ... effect ...

type ... effect ...

judgment ... judgment ...

rule ... type ...

statement
(function type)
type constructor
a set of values: type

$S : \Sigma \rightarrow \Sigma$

base type ... statement ...

a set of all states $\Sigma \rightarrow \Sigma$ ...

state ... state ...

Subject:
Year: Month: Date:

$[x := a]^l : \Sigma \to \Sigma$

$[skip]^l : \Sigma \to \Sigma$

axioms

$$\frac{S_1 : \Sigma \to \Sigma \qquad S_2 : \Sigma \to \Sigma}{S_1 ; S_2 : \Sigma \to \Sigma}$$

$$\frac{S_1 : \Sigma \to \Sigma \qquad S_2 : \Sigma \to \Sigma}{\text{if } [b]^l \text{ then } S_1 \text{ else } S_2 : \Sigma \to \Sigma}$$

$$\frac{S : \Sigma \to \Sigma}{\text{while } [b]^l \text{ do } S : \Sigma \to \Sigma}$$

proper rules

BARAN

Subject: PA

Year: 99 Month: M Date: 9

Annotated Type System , Effect System

- Effect System: $S : \Sigma \xrightarrow{\varphi} \Sigma$

- Annotated Type System: $S : \Sigma_1 \rightarrow \Sigma_2$

Subject:
Year: Month: Date:

Annotated Base Type:

Reaching Definitions

$S : RD_1 \longrightarrow RD_2$

$RD_1, RD_2 \in \mathcal{P}(Var \times Lab)$

rule judgment Sense

$init(S)$ ⟶ label of entry block for S

$final(S)$ ⟶ label of exit blocks for S

final init statement

$RD_1 = RD_{entry}(init(S))$: RD_1

$RD_2 = \bigcup \{ RD_{exit}(l) \mid l \in final(S) \}$: RD_2

union

meaning

Subject: PA

Year: 1394 Month: ۴ Date: ۵

$S : RD_1 \to RD_2$ judgment

$$[x := a]^{l'} : RD \to \left(RD \setminus \{(a,l) \mid l \in Lab\}\right) \cup \{(x,l')\}$$

$$[skip]^{l'} : RD \to RD$$

$$\frac{S_1 : RD_1 \to RD_2 \qquad S_2 : RD_2 \to RD_3}{S_1 ; S_2 : RD_1 \to RD_3} \quad (SEQ)$$

$$\frac{S_1 : RD_1 \to RD_2 \qquad S_2 : RD_1 \to RD_2}{\text{if } [b]^l \text{ then } S_1 \text{ else } S_2 : RD_1 \to RD_2}$$

↓

تعریف RD ها

LRD

if

branch

restrictive

$$\frac{S : RD \to RD}{\text{while } [b]^l \text{ do } S : RD \to RD}$$

↓

entry , exit

BARAN

۳۳

Subject:
Year: Month: Date:

$$S : RD_2 \rightarrow RD_3 \quad RD_1 \subseteq RD_2 \quad RD_3 \subseteq RD_4$$
Subsumption

$$S : RD_1 \rightarrow RD_4$$

(subset) کمتر از
(superset) بیشتر از

اگر S_1 و S_2 دو برنامه باشند، اگر S_2 subsumption S_1 باشد یعنی
خروجی S_2 دقیق‌تر است.

subsumption به معنی meaning یکسان یا نزدیک به هم

meaning دقیق به صورت safe approx. بیان می‌شود:

$$RD_1 \subseteq RD_{entry}(init(S))$$

$$RD_2 = \bigcup \{ \forall \ell \in final(S).\ RD_{exit}(\ell) \subseteq RD_2 \mid \ell \in final(S) \}$$

Example –

$$[y := x]^1 ; [z := 1]^2 ; \text{while } [y > 1]^3 \text{ do } ([z := z * y]^4 ; [y := y - 1]^5) ; [y := 0]^6$$

given:
$RD_f = \{(x,?), (y,1), (y,5), (z,2), (z,4)\}$

BARAN

Subject: PA

Year: 94 Month: 15 Date: 0

$[z := 2*y]^4 : RD_f \longrightarrow \{(x,?),(y,1),(y,5),(z,4)\}$

$\{(x,?),(y,1),(y,5),(z,4),(z,2)\}$

$[y := y-1]^5 : \{(x,?),(y,1),(y,5),(z,4)\}$

$\longrightarrow \{(x,?),(y,5),(z,4)\}$

$([z := 2*y]^4 ; [y := y-1]^5) : RD_f \longrightarrow RD_f$

Subsumption, SEQ

$\text{while } [y>1]^3 \text{ do } ([z := 2*y]^4 ; [y := y-1]^5) : RD_f \longrightarrow RD_f$

$[y := x]^1 : \{(x,?),(y,?),(z,?)\} \longrightarrow \{(x,?),(y,1),(z,?)\}$

$[z := 1]^2 : \{(x,?),(y,1),(z,?)\} \longrightarrow \{(x,?),(y,1),(z,2)\}$

$[y := 0]^6 : RD_f \longrightarrow \{(x,?),(y,6),(z,2),(z,4)\}$

$: \{(x,?),(y,?),(z,?)\} \longrightarrow \{(x,?),(y,6),(z,2),(z,4)\}$

BARAN

equational

Subject:

Year: Month: Date:

PA1961319l3.pdf ← ترجمه کنید اول فردا

manual ... Constraint-based ... automatic ...

Annotated Type Constructors:

Type Constructor ... Type Computation ...

arrow type ←

$S : \Sigma \xrightarrow[RD]{X} \Sigma$: judgment ...

(annotate ... arrow ...)

Effects ...

S ... assign ... X ... definitely ... S ... RD ... Reaching Definition ... S ... might

$[x := a]^l : \Sigma \xrightarrow[\{(x,l)\}]{\{x\}} \Sigma$

BARAN

Subject: PA
Year: ۱۳۹۴ Month: ۱۲ Date: ۷

$$[skip]^{\ell} : \Sigma \xrightarrow[\emptyset]{\emptyset} \Sigma$$

$$S_1 : \Sigma \xrightarrow[RD_1]{X_1} \Sigma \qquad S_2 : \Sigma \xrightarrow[RD_2]{X_2} \Sigma$$

(*) مشترک

$$S_1 ; S_2 : \Sigma \xrightarrow[(RD_1 \setminus X_2) \cup RD_2]{X_1 \cup X_2} \Sigma$$

X_2 بدون X_1 ← برچسب‌های موجود اول یکا

$$S_1 : \Sigma \xrightarrow[RD_1]{X_1} \Sigma \qquad S_2 : \Sigma \xrightarrow[RD_2]{X_2} \Sigma$$

$$\text{if } [b]^{\ell} \text{ then } S_1 \text{ else } S_2 : \Sigma \xrightarrow[RD_1 \cup RD_2]{X_1 \cap X_2} \Sigma$$

$$S : \Sigma \xrightarrow[RD]{X} \Sigma$$

$$\text{while } [b]^{\ell} \text{ do } S : \Sigma \xrightarrow[RD]{\emptyset} \Sigma$$

عملیات تکرار S نمی‌شود

اجرا دقیقا صفر بار نمی‌شود.

(if نمی‌شود)

$$S : \Sigma \xrightarrow[RD]{X} \Sigma \qquad X' \subseteq X \quad RD' \supseteq RD \quad \text{Subsumption}$$

$$S : \Sigma \xrightarrow[RD']{X'} \Sigma$$

توجه: Syntax برای Annotated Type Constraints نیست.

BARAN

۴۷

Subject: ابزارهای آشنایی با دارایی و کاربرد آن

Year: Month: Date:

سال ... ← ... (وجه ادامه دارد)

Subject
Date

$$\frac{S_1 : \sum \xrightarrow[RD_1]{X_1} \sum \qquad S_2 : \sum \xrightarrow[RD_2]{X_2} \sum}{\text{if } [b]^l \text{ then } S_1 \text{ else } S_2 : \sum \xrightarrow[RD_1 \cup RD_2]{X_1 \cap X_2} \sum}$$

تعریف این است که قطعاً assign های شود بین اشتراک می‌گیریم.

آنالیز safe ← ممکن است برسه.

$$\frac{S : \sum \xrightarrow[RD]{X} \sum}{\text{while } [b]^l \text{ do } S : \sum \xrightarrow[RD]{\emptyset} \sum}$$

X: definitely...
RD: might...

$$\frac{S : \sum \xrightarrow[RD]{X} \sum}{S : \sum \xrightarrow[RD']{X'} \sum} \quad \text{if } X' \subseteq X \text{ and } RD \subseteq RD'$$

مجموعه بزرگتر — مجموعه کوچکتر

تمرین: انواعهای مختلف برای type system.

(*) $[y := x]^1 ; [z := 1]^2 ; \text{while } [y > 1]^3 \text{ do } ([z := z * y]^4 ; [y := y - 1]^5) ; [y := 0]^6$

$$[z := z * y]^4 : \sum \xrightarrow[\{(z,4)\}]{\{z\}} \sum$$

$$[y := y - 1]^5 : \sum \xrightarrow[\{(y,5)\}]{\{(y)\}} \sum$$

$$[z := z * y]^4 ; [y := y - 1]^5 : \sum \xrightarrow[\{(z,4),(y,5)\}]{\{z,y\}} \sum$$

$$\text{while } [y > 1]^3 \text{ do } ([z := z * y]^4 ; [y := y - 1]^5) : \sum \xrightarrow[\{(z,4),(y,5)\}]{\emptyset} \sum$$

$$\vdots$$

$$(*) : \sum \xrightarrow[\{(y,6),(z,2),(z,4)\}]{\{y,z\}} \sum$$

reaching definition → {may}

Subject
Date

Effect systems :

Judgment ها ترکیبی از type و effect است.

در این جایگاه زبان functional معرفی می‌کنیم که آنالیزی می‌شود call شدن یا apply شدن functionها

متغیرها (x)

$fn\ x \Rightarrow e$; function abstraction

(π: name) ← x

$e_1\ e_2$: application

این تایپ touring complete است.

church-touring thesis . نظریه‌ی محاسبه‌پذیری { touring ← in principle
λ calculus ←

$\Gamma \vdash e : \tau$

$\Gamma \vdash x : \tau_x$ if $\Gamma(x) = \tau_x$

$$\frac{\Gamma[x \mapsto \tau_x] \vdash e : \tau}{\Gamma \vdash fn\ x \Rightarrow e : \tau_x \to \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau}$$

example: روی این تایپ‌ها، effect ها می‌توانند آنالیز انجام دهند.

$(fn_X\ x \Rightarrow x)\ (fn_Y\ y \Rightarrow y)$

تعریف تابع

$[y \mapsto int] \vdash y : int$

$[\,] \vdash fn_Y\ y \Rightarrow y : int \to int$

Subject
Date

$[x \mapsto int \to int] \vdash x : int \to int$

$[] \vdash fn_x\ x \Rightarrow x : (int \to int) \to (int \to int)$

① ②

$[] \vdash (fn_x\ x \Rightarrow x)(fn_y\ y \Rightarrow y) : int \to int$

Call - Tracking Analysis:

For each subexpression, which function abstractions may be applied during its evaluation.

effect استفاده می‌شود
effect: مجموعه نام‌هایی توابعی که هنگام evaluation آن subexpression صدا زده می‌شود.
apply می‌شود.

$int \xrightarrow{\{x\}} int$ ← انتظار داریم برای این function
چیزی بیرون بیاید.

Annotated types
- base types $\hat{\tau}$
- $\hat{\tau}_1 \xrightarrow{\varphi} \hat{\tau}_2$

effect: The names of the function abstractions that we might applying a function of this types

$\Gamma \vdash e : \hat{\tau}\ \&\ \varphi$
- annotated type
- effect of evaluating
توابعی که ممکن است apply شوند.

Subject
Date

Type & effect systems:

$$\hat{\Gamma} \vdash x : \hat{\tau}_x \,\&\, \emptyset \quad \text{if} \quad \hat{\Gamma}(x) = \hat{\tau}_x$$

$$\frac{\hat{\Gamma}\,[x \mapsto \hat{\tau}_x] \vdash e : \hat{\tau} \,\&\, \varphi}{\hat{\Gamma} \vdash \text{fn}_\pi\, x \Rightarrow e : \hat{\tau}_x \xrightarrow{\varphi \text{ of } \pi} \hat{\tau} \,\&\, \emptyset}$$

آنهایی که عمل است apply شوند
آنهایی که هنوز و نقطه آنهایی که
در body است
هستند.

چون هنوز abstraction است
پس apply نمی شوند (∅ می شود)

$$\frac{\hat{\Gamma} \vdash e_1 : \hat{\tau}_2 \xrightarrow{\varphi} \hat{\tau} \,\&\, \varphi_1 \qquad \hat{\Gamma} \vdash e_2 : \hat{\tau}_2 \,\&\, \varphi_2}{\hat{\Gamma} \vdash e_1 e_2 : \hat{\tau} \,\&\, \varphi \cup \varphi_1 \cup \varphi_2}$$

e_2 و e_1 باید apply e_2 و e_1 باید evaluate برای اینکه صدق کند

Example:

$$[\,] \vdash \text{fn}_{\{2\}}\, y \Rightarrow y : \text{int} \xrightarrow{\{2\}} \text{int} \,\&\, \emptyset$$

$$[\,] \vdash \text{fn}_{\{x\}}\, x \Rightarrow x : (\text{int} \xrightarrow{\{2\}} \text{int}) \xrightarrow{\{x\}} (\text{int} \xrightarrow{\{2\}} \text{int}) \,\&\, \emptyset$$

$$[\,] \vdash (\text{fn}_{\{x\}}\, x \Rightarrow x)\ (\text{fn}_{\{2\}}\, y \Rightarrow y) : \text{int} \xrightarrow{\{2\}} \text{int} \,\&\, \{x\}$$

Subject: PA (Program Analysis)
Date 1394/12/14

SAT and SMT:

equational

Syntactic

(set)

fixed Point

(NPC)

SAT (Satisfiability)

(Propositional Logic)

(satisfy)

$I \models \varphi$

interpretation

valid

(Computational Logic)

Subject:
Date

Analysis, Verification ... SAT ...

... SAT ...

↓

Complete, Sound

(Quantifier-free) ... fragment ...

implication formulas

FOL: decidable, ... fragment ...

(First-order Logic)

[generality]: (Satisfiability Modulo Theory) SMT

(uninterpreted) ... Predication ... Function ...

... Int ...

fragment ... Scope

↳ decide ... Computationally ...

SMT, SAT ... developing a feel ...

(Calculation of Computing)

Subject: PA
Date ۹۹/۱۲/۱۲

First order Logical Theory:

(First order SMT Theory)

formulas: a set of functions, predicates, and constants

FALSE, TRUE co-domain

(I) Signature (SMT Theory)

(A) axiom

(meaning)

interpretation

Subject:

Date 1394/12/14

fragment زیر مجموعه‌ای از منطق مرتبه اول است که ...

decidable ... algorithmic

Theory of Equality:

Σ_E : - Any function (uninterpreted), predicate and constant
- The predicate '='

A_E :

$\forall x . x = x$

for all things

$\forall x, y . x = y \rightarrow y = x$

$\forall x, y, z . x = y \wedge y = z \rightarrow x = z$

$\forall x_1, x_2, \dots, x_n, y_1, \dots, y_n . x_1 = y_1 \wedge x_2 = y_2 \wedge \dots \wedge x_n = y_n$

$\rightarrow f(x_1, \dots, x_n) = f(y_1, \dots, y_n)$

if f is a function

$\forall x_1, \dots, x_n, y_1, \dots, y_n . x_1 = y_1 \wedge \dots \wedge x_n = y_n \rightarrow p(x_1, \dots, x_n) = p(y_1, \dots, y_n)$

if p is a predicate

Subject: PA
Date
۱۴۰۲/۱۲/۱۲

T_E is undecidable ← اثبات شود T_E تصمیم‌پذیر است، ...

... Valid ...

61 ... QFF of T_E ...

(quantifier-free)

SMT-Solver ... Satisfiability ...

Example -

$$a=b \wedge b=c \rightarrow g(f(a),b) = g(f(c),\hat{a})$$

... proof ... axiom ...

Theory of Peano Arithmetic:

Σ_{PA} :
- Constants: 0, 1
- Binary functions: +, *
- Predicate: =

... domain ...

A_{PA} :

$\forall x. \neg(x+1 = 0)$ ... least element ...

$\forall x. x+0 = x$

$\forall x. x*0 = 0$

P4PCO

Subject:
Date

$$\forall x,y.\ x+1=y+1 \longrightarrow x=y$$

$$\forall x,y.\ x+(y+1)=(x+y)+1$$

$$\forall x,y.\ x*(y+1)=x*y+x$$

$$F[0] \wedge (\forall x.\ F[x] \rightarrow F[x+1]) \longrightarrow \forall x.\ F[x]$$

اکسیوم‌های این تئوری axiom

undecidable ، کوئنتیفایر QFF

SMT Theory

SMT_Solver

Symbolic Execution:

Syntactic

Symbolic

(Abstract Interpr...) abstract interpreter

Subject PA

Date ۹۹/۱۲/۱۴

Symbolic Execution ← Abstract Interpretation ← Testing

Symbolic Store and Path Constraint

Symbolic state

Symbolic Store $\sigma_s : Var \rightarrow Sym$

(memory)

e.g. $\sigma_s : x \mapsto x_0 , y \mapsto y_0$

$z = x + y$

$x \mapsto x_0 , y \mapsto y_0 , z \mapsto x_0 + y_0$

FOL Theory ← Path Constraint

quantifier

decidable

branch ← taken

if $(x > y+1)$ { 5: ... }

$\sigma_s = x \mapsto x_0 , y \mapsto y_0$

$\sigma_s : x \mapsto x_0 , y \mapsto y_0$

$pct = x_0 > 10$

$Pct : x_0 > 10 \wedge x_0 > y_0 + 1$

Subject
Date

Example -

```
int twice (int v) {
    return 2*v;
}
void test (int x, int y) {
    z = twice(y);
    if (x == z) {
        if (x > y+10) {
            ERROR
        }
    }
}
int main() {
    x = read();
    y = read();
    test(x, y);
}
```

$\sigma_s: x \mapsto x_0, y \mapsto y_0, z \mapsto 2y_0$ (۲)
pct: TRUE

σ_s: the same as (۲) / pct: $x_0 = 2y_0$ (۳)

σ_s: the same, pct: $x_0 = 2y_0 \wedge x_0 > y_0 + 10$ (۵)

(۴) σ_s: The same as (۲)
pct: $x_0 \neq 2y_0$

$\sigma_s: x \mapsto x_0, y \mapsto y_0$
Sym. store
(۱)

i) ... unsatisfiable ... path constraint ... (pct)

... pct ... SMT_Solver ... pct ... (۵) ...

... $x_0 = 40$, $y_0 = 20$ ... ERROR

... trigger

Subject: PA

Year: 99 Month: 11 Date: 15

: Symbolic Executing

path condition ... branch condition ...

path Explosion

twice ...

nonlinear SMT Solver ...

linear

loop ...

bounded ... under approximation

loop invariants

Symbolic Execution ... Concrete Testing

Concolic Execution (one-two)

static analysis

BARAN.

Subject:

Year: Month: Date:

~~Bisimulation~~ & Milner ~~=~~ appendix & Induction & Co-induction & order Thry : ... Fixed Point Thry

and co-induction

Mathematical Preliminaries:

A partial ordering is a relation $\sqsubseteq : L \times L \to \{true, false\}$

that is reflexive, antisymmetric and transitive.

A partially ordered set (poset), is written $(L, \sqsubseteq)$, is a

set L equipped with a partial ordering $\sqsubseteq$.

$l_2 \sqsupseteq l_1$ means $l_1 \sqsubseteq l_2$

(square precedes)

A subset y of L has $l \in L$ as an upper bound

if $\forall l' \in y . l' \sqsubseteq l$. Similarly, y has $l \in L$ as an lower

bound if $\forall l' \in y . l' \sqsupseteq l$.

A least upper bound of $y \subseteq L$ is an upper bound of

y having relation with all upper bounds of y.

Similarly, a greatest lower bound of $y \subseteq L$ is a lower bound

with $\sqsupseteq$ relation with all lower bounds of y.

BARAN

Subject PA
Date ۱۳۹۷/۱۲/۱۴

$\bigsqcup y$ least upper bound of y (lub)
(Supremum)
join
(square cup)

$\bigsqcap y$ greatest lower bound of y (glb)
(infimum)
meet
(square cap)

Definition- A complete lattice $L = (L, \sqsubseteq) = (L, \sqsubseteq, \sqcup, \sqcap, \top, \bot)$ is a poset $(L, \sqsubseteq)$ such that all subsets of L have least upper bounds and greatest lower bounds. Furthermore,

$\bot = \bigsqcup \emptyset = \bigsqcap L$ and $\top = \bigsqcup L = \bigsqcap \emptyset$

(*) poset یک مجموعه ی جزئی مرتب است که هر زیرمجموعه ی آن دارای lub و glb باشد.

Example- $L = (\wp(S), \subseteq)$
any set ↳ subset relation ↳ even infinite set

(اجتماع هر مجموعه از اعضای مجموعه ی اولیه، خود عضوی از مجموعه ی اولیه است.)

$\bigcup y = \{y \mid \exists x \in y \,.\, y \in X\}$

$\bigcap y = \{y \mid \forall x \in y \,.\, y \in X\}$

PqPCO

۹

Subject
Date

Lemma - For a partially ordered set $L = (L, \sqsubseteq)$, the claims

1) L is a complete lattice,

2) Every subset of L has a least upper bound,

3) Every subset of L has a greatest lower bound

are equivalent.

Proof - $Y \subseteq L$

$$\sqcap Y = \sqcap \{ l \in L \mid \forall l' \in Y.\ l \sqsubseteq l' \}$$

A Moore family is a subset Y of a complete lattice $L = (L, \sqsubseteq)$ that is closed under greatest lower bounds.

$$\forall y' \subseteq y.\ \sqcap y' \in y$$

least element
greatest element

Subject PA

Date ۱۳۹۴,۱۲,۱۴

Properties of function :

A function $f : L_1 \to L_2$ between partially ordered sets $L_1 = (L_1, \sqsubseteq_1)$ and $L_2 = (L_2, \sqsubseteq_2)$ is surjective (onto) (epic) if $\forall l_2 \in L_2 . \exists l_1 \in L_1 . f(l_1) = l_2$.

And, it is injective (one-to-one or monic) if

$$\forall l, l' \in L_1 . f(l) = f(l') \to l = l'.$$

(و یا به صورتی که poset را از آن بدست آوردیم)

The function f is monotone (or isotone or order preserving) if $\forall l, l' \in L_1 . l \sqsubseteq_1 l' \to f(l) \sqsubseteq_2 f(l')$.

Subject

Date

Subject PA
Date ۹۹/۱۲/۱۹

Properties of functions:

$(L_1, \sqsubseteq_1)$ $(L_2, \sqsubseteq_2)$

I. onto ⟶ f تابعی پوشا از L_1 به L_2 است، یعنی ...

II. one-to-one ⟶ ... یک‌به‌یک است.

III. monotone

$$l_1 \sqsubseteq_1 l_2 \Rightarrow f(l_1) \sqsubseteq_2 f(l_2)$$

order-preserving

IV. additive (join morphism)

$$\forall l_1, l_2 \in L_1 . f(l_1 \sqcup_1 l_2) = f(l_1) \sqcup_2 f(l_2)$$

(Poset ...) ... lub ...

... lattice ...

V. multiplicative (meet morphism)

$$\forall l_1, l_2 \in L_1 . f(l_1 \sqcap_1 l_2) = f(l_1) \sqcap_2 f(l_2)$$

VI. completely additive

$$\forall y \subseteq L_1 . f(\sqcup_1 y) = \sqcup_2 \underbrace{\{f(l') \mid l' \in y\}}_{f(y)} \text{ whenever } \sqcup_1 y \text{ exists}$$

VII. Similarly, completely multiplicative

Subject
Date

IX. affine

$$\forall\, Y \subseteq L_1 \cdot Y \neq \emptyset \cdot f(\bigsqcup_1 Y) = \bigsqcup_2 \{f(x) \mid x \in Y\}$$

whenever $\bigsqcup_1 Y$ exists.

X. strict

~~f is strict and~~ $f(\bot_1) = \bot_2$

↙

least element of 1

strict , affine f : Completely additive

Lemma – If $L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ and $M = (M, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ are complete lattices and M is finite then the three conditions

(i) $\gamma : M \to L$ is monotone,

(ii) $\gamma(\top) = \top$, and

(iii) $\gamma(m_1 \sqcap m_2) = \gamma(m_1) \sqcap \gamma(m_2)$ whenever $m_1 \not\sqsubseteq m_2$ ~~and~~ $m_2 \not\sqsubseteq m_1$,

are jointly equivalent to $\gamma : M \to L$ being completely multiplicative.

Subject PA

Date ۹۷/۱۲/۱۹

Proof

completely multiplicative

$m_1 \sqsubseteq m_2 \Rightarrow f(m_1 \sqcap m_2) = f(m_1) = f(m_1) \sqcap f(m_2)$

monotonic

$\Rightarrow f(m_1) \sqsubseteq f(m_2)$

Top

completely multiplicative

$m_2 \sqsubseteq m_1$ ، $m_1 \sqsubseteq m_2$

multiplicative

monoton

$m_1 \sqsubseteq m_2 \Rightarrow r(m_1) \sqsubseteq r(m_2)$

$\Rightarrow r(m_1 \sqcap m_2) = r(m_1) = r(m_1) \sqcap r(m_2)$

r is multiplicative

(ii), (iii) $\Rightarrow r(m_1 \sqcap m_2) = r(m_1) \sqcap r(m_2)$

$M' \subseteq M$

$r(\sqcap^{M'}) = \sqcap r(M') =$

$\sqcap \{ r(m) \mid m \in M' \}$

PqPCO

(Transferring Induction)

Subject

Date

if $|M'| = 0$ $(M' = \emptyset)$ then $\gamma(\Pi\emptyset) = \Pi\emptyset$

T T

if $M' = M'' \cup \{m''\}$

$$\gamma(\Pi M') = \gamma((\Pi M'') \sqcap m'') =$$

multiplicative

$$\gamma(\Pi M'') \sqcap \gamma(m'') =$$

$$\Pi\gamma(M'') \sqcap \gamma(m'') = \Pi\gamma(M')$$

Lemma – A function $f : (\mathcal{P}(D), \subseteq) \to (\mathcal{P}(E), \subseteq)$ is affine iff there exists a function $\varphi : D \to \mathcal{P}(E)$ and an element $\varphi_\emptyset \in \mathcal{P}(E)$ s.t. $f(Y) = \bigcup \{\varphi(d) \mid d \in Y\} \cup \varphi_\emptyset$

The function f is completely additive iff additionally $\varphi_\emptyset = \emptyset$.

Subject PA
Date ۱۳۹۹/۱۲/۱۹

Construction of Complete Lattices:

- Cartesian Product:

$L_1 = (L_1, \sqsubseteq_1)$ and $L_2 = (L_2, \sqsubseteq_2)$ posets

$L = (L, \sqsubseteq)$

$L = \{(l_1, l_2) \mid l_1 \in L_1, l_2 \in L_2\}$ ∧

$(l_{11}, l_{21}) \sqsubseteq (l_{12}, l_{22})$ iff $l_{11} \sqsubseteq_1 l_{12} \wedge l_{21} \sqsubseteq_2 l_{22}$.

$\bigsqcup Y = (\bigsqcup_1 \{l_1 \mid \exists l_2 . (l_1, l_2) \in Y\}, \bigsqcup_2 \{l_2 \mid \exists l_2 . (l_1, l_2) \in Y\})$

$\bot = (\bot_1, \bot_2)$ and Similarly for other elements ...

- Total Function Space:

$L_1 = (L_1, \sqsubseteq_1)$

↓

$L = (L, \sqsubseteq)$ (S is a set)

$L = \{f : S \to L_1 \mid f$ is a total function$\}$ ∧ $f \sqsubseteq f'$ iff $\forall s \in S . f(s) \sqsubseteq_1 f'(s)$.

Subject
Date

$(Y \subseteq L)$

$$\sqcup Y = \lambda s.\ \sqcup_1 \{f(s) \mid f \in Y\}$$

- Monotone Function Space:

$L_1 = (L_1, \sqsubseteq_1)$ and $L_2 = (L_2, \sqsubseteq_2)$ posets

$L = (L, \sqsubseteq)$

$L = \{f : L_1 \to L_2 \mid f \text{ is a monotone function}\}$

$f \sqsubseteq f'$ iff $\forall l_1 \in L_1 .\ f(l_1) \sqsubseteq_2 f(l_1)$

$$\sqcup Y = \lambda l_1 .\ \sqcup_2 \{f(l_1) \mid f \in Y\}$$

Chains:

beat solution: approximate solution, chain

Subject PA

Date 1394, 11, 19

(Calculus → Mathematical Analysis → Topology)

(metric space)

تعریف: Chain

$(L, \sqsubseteq)$ a poset

A subset $Y \subseteq L$ is a chain if

$$\forall l_1, l_2 \in Y.\ (l_1 \sqsubseteq l_2) \vee (l_2 \sqsubseteq l_1)$$

(... totally ordered set ... poset ...)

- A sequence $(l_n)_n = (l_n)_{n \in \mathbb{N}}$ of elements in L is an

ascending chain if $n \leq m \Rightarrow l_n \sqsubseteq l_m$

(descending) ($\exists$)

(... Category Theory, order Theory ...)

(... Continuous partial order ...)

(CPO)

writing $(l_n)_n$ also for $\{l_n \mid n \in \mathbb{N}\}$

- A sequence $(l_n)_n$ eventually stabilizes (... iff

$\exists n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}.\ n \geq n_0 \Rightarrow l_n = l_{n_0}$

Subject
Date

induction, co-induction
greatest fixed point

- We write $\bigsqcup_n l_n$ for $\bigsqcup\{l_n \mid n \in \mathbb{N}\}$

$\bigsqcap_n l_n$ for $\bigsqcap\{l_n \mid n \in \mathbb{N}\}$

Ascending Chain Condition :
(Descending)

- A $(L, \sqsubseteq)$ has finite height iff all chains are finite.

It has height at most h if all chains contains at most $h+1$ elements.

It has finite height h if additionally there is a chain with $h+1$ elements.

- The partially ordered set $(L, \sqsubseteq)$ satisfies the Ascending (Descending) Chain Condition iff all ascending chains eventually stabilize. (descending)

Subject PA (تحلیل ایستا)
Date ۱۴۰۲/۱۲/۲۱

Lemma - A poset $L=(L, \sqsubseteq)$ has finite height iff it satisfies both the Ascending and Descending chain conditions.

Proof -

Left-to-Right :

$(L, \sqsubseteq)$ has finite height

$\Rightarrow$ an ascending chain $(l_n)_n$ is finite $\Rightarrow$ $(l_n)_n$ eventually stabilize

Right-to-Left :

$(L, \sqsubseteq)$ satisfies ascending and descending chain conditions

and $Y \subseteq L$ is a chain.

$Y \neq \emptyset$ (nonempty)

each nonempty subset Y' of Y contains a least element.

constructively :

Subject

Date

به طور استقرایی عضوهای دنباله‌ی (l_n) را به صورتی می‌سازیم که l_{n+1} از l_n بزرگتر نباشد:

l_0 را برابر با least element مجموعه‌ی L در نظر می‌گیریم:

l_{n+1} را برابر با least element مجموعه‌ی $\{l \in L \mid l \leq l_n \wedge l \neq l_n\}$ در نظر می‌گیریم.

پس دنباله‌ی (l_n) به ازای یک n به بزرگترین عضو L می‌رسد و بعد از آن ثابت می‌ماند. $\Leftarrow$ eventually stabilize

از روی این، حالا می‌توانیم ثابت کنیم که هر زنجیر صعودی در L دارای بزرگترین عضو است.

ادامه‌ی اثبات را شما بنویسید:

$(l_n)_n$ in L (Ascending chain)

l_n is chosen as the least element of $L \setminus \{l_0, \ldots, l_{n-1}\}$

$\downarrow$

بزرگترین عضو L

به ازای هر n که این بزرگترین عضو L در دنباله‌ی (l_n) نیست، l_{n+1} را برابر با least element مجموعه‌ی $L \setminus \{l_0, \ldots, l_n\}$ در نظر می‌گیریم.

این عضو حتماً وجود دارد.

این زنجیر صعودی $\Leftarrow$ eventually stabilize

$\downarrow$

پس هر زنجیر صعودی

Subject PA
Date 1394/12/21

when $Y \setminus \{l_0, \ldots, l_{n-1}\} = \emptyset$, $l_n = l_{n-1}$

... $Y \setminus \{l_0, \ldots, l_{n-1}\}$ ...

... Y is finite ⟸ ... Y ...

Ascending chain (descending) condition, finite height, Cartesian Product ... lattice Construction

...

Total Space Function ... finite height ... L ...

... S ...

... Monotone function ...

Lemma - For a poset $(L, \sqsubseteq)$ the conditions:

(1) L is a complete lattice satisfying the Ascending chain condition

(2) L has a least element, $\bot$, and binary least upper bounds and satisfies the Ascending chain condition,

are equivalent.

Subject
Date

part -

(1) ، (2) واضح است.

(2) ، (1) : ثابت می‌کنیم هر زیرمجموعه‌ی $\downarrow$ dcpo دارای

که $\bigsqcup Y \in Y$ ، $Y = \emptyset$ ، $\bigsqcup \emptyset = \bot$ بزرگترین عضو

dcpo است.

اگر Y یک زیرمجموعه‌ی $\downarrow$ دلخواه از dcpo P باشد، آنگاه Y یک زیرمجموعه‌ی $\downarrow$ dcpo

خواهد بود.

از Y یک زیرمجموعه‌ی $\downarrow$ ناتهی می‌گیریم:

$(l_n)_n$ of elements of L

s.t. l_0 be an arbitrary element y_0 of Y.

$l_{n+1} = l_n$ if $\forall y \in Y \cdot y \subseteq l_n$

$l_{n+1} = l_n \sqcup y_{n+1}$ for $y_{n+1} \in Y, y_{n+1} \not\subseteq l_n$ otherwise

(زیرمجموعه‌ی l_{n+1} در Y نباشد آنگاه l_{n+1} در L است)

اگر $(l_n)_n$ stabilize شود آنگاه $\forall y \in Y \cdot y \subseteq l_n$

$\downarrow$

یعنی $\bigsqcup Y \in Y$ می‌شود

$\{y_0, y_1, \dots, y_n\} \subseteq Y$

Subject PA
Date ۱۳۹۷,۱۲,۲۱

l_0 is the lub of Y ← ... l_0 ... $(y_0$ ...

Lemma - For a complete lattice $(L, \sqsubseteq)$ satisfies the ascending chain condition and a total function $f: L \to L$, the condition,

(1) f is additive, i.e., $\forall l_1, l_2 . f(l_1 \sqcup l_2) = f(l_1) \sqcup f(l_2)$.

(2) f is affine, i.e., $\forall Y \subseteq L . Y \neq \emptyset . f(\bigsqcup Y) = \bigsqcup f(Y)$

$\updownarrow$

$\{f(l) \mid l \in Y\}$

are equivalent, and in both cases f is a monotone function.

... (1) ... (2) ... (1) ... f ... :(2) ⇒ (1)

- Y finite $\Rightarrow Y = \{y_1, \dots, y_n\}$

$$f(\bigsqcup Y) = f(y_1 \sqcup y_2 \sqcup \dots \sqcup y_n) = f(y_1) \sqcup \dots \sqcup f(y_n) \sqsubseteq \bigsqcup f(Y)$$

Subject
Date

- Y infinite $\Rightarrow \bigsqcup Y = \ell_n = y_0 \sqcup y_1 \sqcup \dots \sqcup y_n$

به ℓ_n ای میرسیم (متناهی)

پس ما متناهی‌ش y رو میگیریم:

$$f(\bigsqcup Y) \sqsubseteq \bigsqcup f(Y)$$

اثبات: $\bigsqcup f(Y) \sqsubseteq f(\bigsqcup Y)$ هم باید

$$f(\bigsqcup Y) \sqsupseteq \bigsqcup \{f(\ell) \mid \ell \in Y\}$$

$$\forall \ell \in Y.\ \ell \sqsubseteq \bigsqcup Y \Longrightarrow \forall \ell \in Y.\ f(\ell) \sqsubseteq f(\bigsqcup Y)$$

$$\Rightarrow \bigsqcup f(\ell) \sqsubseteq f(\bigsqcup Y).$$

برای monotone

(اثبات)

Fixed Points:

Consider a monotone function $f: L \to L$ on a complete lattice $(L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$

- A fixed point of f is an element $\ell \in L$ s.t. $f(\ell) = \ell$.

$$Fix(f) = \{\ell \in L \mid f(\ell) = \ell\}$$

Subject PA
Date ۱۳۹۹/۱۲/۲۱

- The function f is reductive at l iff $f(l) \sqsubseteq l$
(pre-fixed point)

$$Red(f) = \{l \in L \mid f(l) \sqsubseteq l\}$$

- The function f is extensive at l iff $l \sqsubseteq f(l)$

$$Ext(f) = \{l \in L \mid f(l) \sqsupseteq l\}$$

- $lfp(f) = \overset{glb}{\sqcap} Fix(f)$
(least fixed point)

$$gfp(f) = \bigsqcup Fix(f)$$

(*) Tarski's Fixed Point Theorem - Let $(L, \sqsubseteq, \sqcup, \sqcap, \top, \bot)$ be a complete lattice. If $f: L \to L$ is a monotone function, then

$$lfp(f) = \sqcap Red(f) \in Fix(f)$$

$$gfp(f) = \bigsqcup Ext(f) \in Fix(f)$$

Subject
Date

Proof - $l_0 = \sqcap Red(f)$

$\rightarrow l_0 \sqsubseteq l$ for all $l \in Red(f)$

f is monotone

$\Rightarrow f(l_0) \sqsubseteq f(l) \sqsubseteq l$ for $l \in Red(f)$

$f(l_0)$ ... $Red(f)$ ... l_0 ... glb ...

(I) $f(l_0) \sqsubseteq l_0$

monotonicity

$f(f(l_0)) \sqsubseteq f(l_0)$

$\Rightarrow f(l_0) \in Red(f)$

$\Rightarrow l_0 \sqsubseteq f(l_0)$ (II)

glb l_0

I, II $\rightarrow l_0 = f(l_0)$

$\sqcap Red(f) \in Fix(f)$

$\sqcap Fix(f) = \sqcap Red(f)$

(III) $Fix(f) \subseteq Red(f) \Rightarrow lfp(f) = \sqcap Fix(f) = l_0$

glb

Subject PA
Date ۱۳۹۶/۱۲/۲۱

⊗ اگر f پیوسته باشد، $(f^n(\bot))_n$ یک زنجیره صعودی است.

این زنجیره همگرا می‌شود به:
$\bot \sqsubseteq f^n(\bot) \sqsubseteq \bigsqcup_n f^n(\bot) \sqsubseteq lfp(f) \sqsubseteq gfp(f) \sqsubseteq f^n(\top) \sqsubseteq \top$.

فرض کنیم $f(\bigsqcup Y) = \bigsqcup f(Y)$ for all $Y \subseteq L$ برقرار است.

(پیوستگی) Continuity (یا: $f(\bigsqcup_n l_n) = \bigsqcup_n f(l_n)$)

مجموعه M را به این صورت تعریف می‌کنیم:

$M = \{\bot, f(\bot), f^2(\bot), \ldots\}$

$f(\bigsqcup M) = \bigsqcup f(M)$

$f(M) = \{f(\bot), f^2(\bot), \ldots\} = M \setminus \{\bot\}$

$\bigsqcup f(M) = \bigsqcup M$

$\bot$ نقشی در $\bigsqcup$ ندارد.

$\Rightarrow \boxed{f(\bigsqcup M) = \bigsqcup M} = lfp(M)$

پیدا کردن اینکه f، additive است یا affine، پس اثبات باید برقرار باشد.

این دو شرط پیوستگی را نشان می‌دهند، آیا هم چی؟

⊗ هر زنجیر صعودی در یک poset، یک سوپریمم در آن دارد ...

۴۹

Subject PA
Date ۱۴۰۴/۱۲/۲۱

⊗ سپ ... شروع از ⊥ ، ... f(⊥) ... برسیم ، ... لحظه ... از ... ما ...

lfp ... (... f ... poset ...).

gfp ... T ... f ... apply ... f ... ...

⊗ ... : ... ...

... (... CPO ...)

⊗ ... lfp ... gfp ... co-inductive ...

...

lfp ↔ inductive definition ↔ algebra

gfp ↔ co-inductive definition ↔ co-algebra

... :

☒ ... ← Sangorin ← ... ← ... ← Solution

☒ ...

Subject PA

Date ۹۸/۱۰/۱۸

Data Flow Analysis :

کمیل های مربوط به جریان داده

* اینجا هدف محاسبه اطلاعاتی درباره چگونگی مقادیر متغیرها (موضوع کلی) در برنامه است.

* تمام کمیل های مطرح شده، in general، همین ایده را دارند.

* انواع کمیل جریان داده داریم مثل Very Busy Expressions / Reaching Definitions / Available Expressions

و Live Variables و ...

* Information Flow Analysis vs. Data Flow Analysis

↓ ↓

مبتنی بر deduction و استنتاج از روی برنامه مبتنی بر داده‌ها و جریان داده

(استدلال با تحلیل) مقادیر متغیرها در نقاط مختلف برنامه

روش‌هایی که اینجا Data Flow Graph، Control Flow Graph، مطرح می‌کنیم برای کمیل استاندارد.

کمیل جریان داده مربوط به یکی از کمیل‌های مختلف در برنامه‌هاست از این قسم، اما کمیل جریان اطلاعات کاربرد

مشخص برنامه دارد که شاخه‌ی خاصی تعریف می‌شود و به خط خاصی این متغیر را می‌رسد.

این عبارت بین این دو اهمیت دارد و از همین متغیر مشخص می‌کند که IFA، از چه قسم معادی

PFA استفاده کند.

Subject
Date

Intra-Procedural Analysis :

در این روش تحلیل، ما فقط به بررسی یک تابع می‌پردازیم و نقاط ورود و خروج آن را بررسی می‌کنیم.

Initial and final labels :

$init : stmt \longrightarrow Lab$

تابع init یک statement را می‌گیرد و برچسب اولین statement داخل آن را برمی‌گرداند.

$init([x := a]^l) = l$

$init([skip]^l) = l$

$init(S_1 ; S_2) = init(S_1)$

$init(\text{if } [b]^l \text{ then } S_1 \text{ else } S_2) = l$

$init(\text{while } [b]^l \text{ do } s) = l$

$final : stmt \longrightarrow \wp(Lab)$

تابع final یک statement را می‌گیرد و مجموعه‌ای از برچسب‌های آخرین statement های اجرا شده را برمی‌گرداند.

$final([x := a]^l) = \{l\}$

$final([skip]^l) = \{l\}$

Subject PA

Date ۱۳۹۲/۱/۱۸

$final(S_1 ; S_2) = final(S_2)$

$final(\text{if } [b]^l \text{ then } S_1 \text{ else } S_2) = final(S_1) \cup final(S_2)$

$final(\text{while } [b]^l \text{ do } S) = \{l\}$

به صورت syntact. این مجموعه‌ها را می‌توان محاسبه کرد.

Blocks:

بلوک‌های مجزا برنامه را می‌گیرند.

$blocks : Stmt \longrightarrow \mathcal{P}(Blocks)$

$blocks([x := a]^l) = \{[x := a]^l\}$

$blocks([skip]^l) = \{[skip]^l\}$

$blocks(S_1 ; S_2) = blocks(S_1) \cup blocks(S_2)$

$blocks(\text{if } [b]^l \text{ then } S_1 \text{ else } S_2) = \{[b]^l\} \cup blocks(S_1) \cup blocks(S_2)$

$blocks(\text{while } [b]^l \text{ do } S) = \{[b]^l\} \cup blocks(S)$

اینجا از ترتیب این تعریف‌ها استفاده کردیم. ترتیب این تعریف‌ها را می‌توان عوض کرد.

Subject
Date

Labels:

لیبل‌ها: مجموعه‌ای از لیبل‌ها را برمی‌گرداند.

$$labels : Stmt \longrightarrow \wp(Lab)$$

$$labels(S) = \{ l \mid [B]^l \in blocks(S) \}$$

Flows and Reverse Flows:

فلوها و فلوهای معکوس: یال‌های جهت‌دار گراف را مشخص می‌کنند.

(گراف: مجموعه زوج‌هایی از مجموعه رئوس و یال‌ها)

$$flow : Stmt \longrightarrow \wp(Lab \times Lab)$$

$$flow([x := a]^l) = \emptyset$$

$$flow([skip]^l) = \emptyset$$

$$flow(S_1; S_2) = flow(S_1) \cup flow(S_2) \cup \{ (l, init(S_2)) \mid l \in final(S_1) \}$$

S_2 ← اتصال آخرین گره‌های S_1 به شروع S_2

$$flow(\text{if } [b]^l \text{ then } S_1 \text{ else } S_2) = flow(S_1) \cup flow(S_2) \cup \{ (l, init(S_1)), (l, init(S_2)) \}$$

$$flow(\text{while } [b]^l \text{ do } S) = flow(S) \cup \{ (l, init(S)) \} \cup \{ (l', l) \mid l' \in final(S) \}$$

Subject PA
Date ۱۳۹۷/۱/۱۸

forward برای تحلیل‌هایی که از backward استفاده می‌کنند، flow را برعکس می‌کنیم.

Reverse flow ← برای تحلیل‌های backward، flow را برعکس می‌کنیم.

$$Flow^R : Stmt \to \wp(Lab \times Lab)$$

$$Flow^R(S) = \{(l, l') \mid (l', l) \in Flow(S)\}$$

The program of interest: برنامه‌ای که می‌خواهیم تحلیل کنیم.

S_* یکی از برنامه‌هایی که می‌خواهیم تحلیل کنیم.

Lab_* ، Var_* ، $Blocks_*$ در برنامه S_* استفاده می‌شوند.

برای تحلیلی که می‌خواهیم بنویسیم، نیاز داریم که بدانیم در هر برچسب چه عبارت‌هایی محاسبه می‌شوند. این را برای کدام عبارت‌ها می‌خواهیم بدانیم:

برنامه‌هایی که می‌خواهیم تحلیل کنیم.

non-trivial arithmetic subexpression: $AExp_*$

عبارت‌های حسابی غیر بدیهی در S_*

در S_*

Variable ، Constant نیستند

$a * b$ ، $a + b$ ، $-a$ ، ...

$AExp(a)$ ← عبارت‌های حسابی غیر بدیهی (non-trivial) در a

$AExp(b)$ ← عبارت‌های حسابی غیر بدیهی (non-trivial) در b

Subject
Date

- the program S_* has isolated entries if

$$\forall l \in Lab.\ (l, init(S)) \notin flow(S_*)$$

- the program S_* has isolated exits if

$$\forall l_1 \in final(S_*).\ \forall l_2 \in Lab.\ (l_1, l_2) \notin flow(S_*)$$

- A statement S is label consistent iff

$$[B_1]^l, [B_2]^l \in blocks(S) \Rightarrow B_1 = B_2$$

- S is uniquely labeled if each label occurs only once.

Subject PA
Date ۱۵۹۷/۱/۱۸

Available Expressions Analysis:

این تحلیل می‌خواهد سوال زیر را جواب بدهد:

For each program point, which expressions must have already been computed and not later modified on all paths to the program point.

(برای استفاده کردن در تعدادی بهینه‌سازی مثل ... و ...)

Example -

$[x := a+b]^1 ; [y := a*b]^2 ; \text{while } [y > a+b]^3 \text{ do}$

$([a := a+1]^4 ; [x := a+b]^5)$

در نقطه 3، $a+b$ یک available exp است، برای ... می‌توانیم ...

زیرا از ... به نقطه 3 دو مسیر وجود دارد و در هیچ کدام a و b عوض نشده‌اند.

از نقطه 3 به نقطه 5 یک مسیر وجود دارد و در آن a و b عوض نشده‌اند.

از نقطه 3 به نقطه 4 یک مسیر وجود دارد و در آن a عوض شده است.

برای تعریف دقیق‌تر این تحلیل، دو مفهوم جدید مهم داریم: kill, generate.

Subject
Date

$$kill_{AE} : Blocks_* \longrightarrow \mathcal{P}(AExp_*)$$

An expression is killed in a block if some variable used in the expression are modified in the blocks.

$$kill_{AE}([x := a]^l) = \{ a' \in AExp_* \mid x \in FV(a') \}$$

$$kill_{AE}([skip]^l) = \{\}$$

$$kill_{AE}([b]^l) = \emptyset$$

no kill, assignment

$$gen_{AE} : Blocks_* \longrightarrow \mathcal{P}(AExp_*)$$

A generated expression is an expression that is evaluated in the block and where none of the variables used in the expression are later modified in the block.

$$gen_{AE}([x := a]^l) = \{ a' \in AExp(a) \mid x \notin FV(a') \}$$

Subject PA
Date ۱۳۹۷/۱۱/۱۸

$$gen_{AE}([skip]^l) = \emptyset$$

$$gen_{AE}([b]^l) = AExp(b)$$

Available Expressions برای محاسبه

$$AE_{entry}(l) = \begin{cases} \emptyset & \text{if } l = init(S_*) \\ \bigcap \{ AE_{exit}(l') \mid (l', l) \in flow(S_*) \} & \text{otherwise} \end{cases}$$

(must) یک عبارت زمانی در دسترس است که در تمام مسیرها محاسبه شده باشد و [b]^l در مسیر قرار نگیرد.

از اطلاعات forward است، یعنی از ابتدای برنامه شروع می‌شود.

$$AE_{exit}(l) = (AE_{entry}(l) \setminus kill_{AE}(B^l)) \cup gen_{AE}(B^l)$$

where $B^l \in Blocks_*(S_*)$

(فقط برای بلوک‌ها تعریف می‌شود)

یعنی عبارت‌هایی که بعد از بلوک در دسترس هستند.

یعنی عبارت‌هایی که قبل از بلوک در دسترس هستند.

برای بدست آوردن بهترین جواب از fixed point استفاده می‌کنیم.

greatest fixed point یعنی بزرگ‌ترین جواب ممکن

[Top مجموعه تهی]

Co-inductive یعنی Available Expressions

از آنجایی که ما به دنبال بزرگترین جواب هستیم در ابتدا همه را ∅ قرار می‌دهیم و سپس کوچک می‌کنیم.

Subject

Date

Data Flow Value ← ...

* Analysis ⊢ must e.g. Available Expression
  ↳ may

* Analysis ⊢ Forward e.g. Available Exp
  ↳ Backward

intra-procedural ...

forward, must ... Available Expression ...

Top (...AE ...)

iterative ... lower approximate, must ...

(∅) bot ... upper approx, may ...

entry (...) exit (...) ... block ...

may/must ... forward/backward ...

Example –

$[z := x+y]^{l}$; while $[true]^{l'}$ do $[skip]^{l''}$

$$AE_{entry}(l) = \emptyset$$

$$AE_{entry}(l') = AE_{exit}(l) \cap AE_{exit}(l'')$$

$$AE_{entry}(l'') = AE_{exit}(l')$$

$$AE_{exit}(l) = AE_{entry}(l) \cup \{x+y\}$$ ← generate

$$AE_{exit}(l') = AE_{entry}(l') \qquad AE_{exit}(l'') = AE_{entry}(l'')$$

Subject PA
Date ۹۷/۱/۲۰

$AE_{entry}(l') = \{x+y\} \cap AE_{entry}(l')$

$AE_{entry}(l') = \emptyset$

$AE_{entry}(l') = \{x+y\}$

Reaching Definitions Analysis:

For each program point, which assignments may have been made and not overwritten, when program execution reaches this point along some path.

Subject
Date

undefined یعنی اسایشگاهی که در زمان ورود ? بود

$$kill_{RD} : Blocks_* \to \wp(Var_* \times Lab_*^?)$$

$$gen_{RD} : Blocks_* \to \wp(Var_* \times Lab_*^?)$$

$$RD_{entry}, RD_{exit} : Lab_* \to \wp(Var_* \times Lab_*^?)$$

این روابط برای forward و may هستند. (Reaching Definitions)

(اسامی مهم، تعریف توابع، معادلات)

$$kill_{RD}([x := a]^l) = \{(x, ?)\} \cup \{(x, l') \mid B^l \text{ is an assignment to } x \text{ in } S_*\}$$

$$kill_{RD}([skip]^l) = \emptyset$$

$$kill_{RD}([b]^l) = \emptyset$$

$$gen_{RD}([x := a]^l) = \{(x, l)\}$$

$$gen_{RD}([skip]^l) = \emptyset$$

$$gen_{RD}([b]^l) = \emptyset$$

Subject PA
Date 1401/1/10

uniquely labeled + isolated entries : ...

$$RD_{entry}(\ell) = \begin{cases} \{(x,?) \mid x \in FV(S_*)\} & \text{if } \ell \in init(S_*) \\ \bigcup \{RD_{exit}(\ell') \mid (\ell',\ell) \in flow(S_*)\} \end{cases}$$

$$RD_{exit}(\ell) = \left(RD_{entry}(\ell) \setminus kill_{RD}(B^\ell)\right) \cup gen_{RD}(B^\ell)$$

$$B^\ell \in blocks(S_*)$$

... least fixed point ... smallest ...

Very Busy Expressions Analysis:

An expression is very busy at the exit from a label if, no matter what path is taken from the label, the expression always be used before any of the variables occurring in it are redefined.

... (backward, must ...)

For each program point, which expressions must be very busy at the exit from the point.

initial, final ...

(backward, must ...)

Subject
Date

(hoisting)

Example -

if $[a>b]^1$ then $([x:=b-a]^2; [y:=a-b]^3)$

else $([y:=b-a]^4; [x:=a-b]^5)$

very busy ، $b-a$ ، $a-b$

$b-a$ ، $a-b$

$$kill_{VB} : Blocks_* \longrightarrow \wp(AExp_*)$$

$$gen_{VB} : Blocks_* \longrightarrow \wp(AExp_*)$$

$$kill_{VB}([x:=a]^l) = \{a' \in AExp_* \mid x \in FV(a')\}$$

redefine

$$kill_{VB}([skip]^l) = \emptyset$$

$$kill_{VB}([b]^l) = \emptyset$$

Subject PA
Date ۱۳۹۷/۱۱/۲۰

$$gen_{VB}([x := a]^l) = AExp(a)$$

$$gen_{VB}([Skip]^l) = \emptyset$$

$$gen_{VB}([b]^l) = AExp(b)$$

$$VB_{exit}(l) = \begin{cases} \emptyset & \text{if } l \in final(S_*) \\ \bigcap \{ VB_{entry}(l') \mid (l', l) \in Flow^R(S_*) \} & \text{otherwise} \end{cases}$$

[isolated exits, uniquely labeled]

$$VB_{entry}(l) = \left(VB_{exit}(l) \setminus kill_{VB}(B^l) \right) \cup gen_{VB}(B^l)$$

$$B^l \in Blocks(S_*)$$

$$VB_{entry}, VB_{exit} : Lab_* \longrightarrow \mathcal{P}(AExp_*)$$

largest set

Subject

Date

Live Variables Analysis :

A variable is live at the exit from a label if there exists a path from the label to a use of the variable that does not redefine the variable.

این یک backward تحلیل است.

For each program point, which variables may be live at the exit from the point.

این برای حذف dead code است.

$$[x := 2]^1 ; [y := 4]^2 ; [x := 1]^3 ;$$

$$(\text{if } [y > x]^4 \text{ then } [z := y]^5 \text{ else } [z := y * y]^6) ; [x := z]^7$$

در خروجی بلوک ۱، x زنده (live) است. چون مسیری از خروجی ۱ وجود دارد که x در آن استفاده شود،

بدون اینکه قبل از آن redefine شود. اما y در خروجی ۱ زنده نیست.

(dead code) elimination

Subject PA

Date ۱۴۹۷/۱۲/۱۵

(smallest set)

backward , may

$kill_{LV} : Blocks_* \to \mathcal{P}(Var_*)$

$gen_{LV} : Blocks_* \to \mathcal{P}(Var_*)$

$kill_{LV}([x := a]^l) = \{x\}$

redefine x

$kill_{LV}([skip]^l) = \{\}$

$kill_{LV}([b]^l) = \{\} = \emptyset$

$gen_{LV}([x := a]^l) = FV(a)$

$gen_{LV}([skip]^l) = \emptyset$

$gen_{LV}([b]^l) = FV(b)$ → use

$LV_{exit}, LV_{entry} : Lab_* \to \mathcal{P}(Var_*)$

isolated exit

$$LV_{exit}(l) = \begin{cases} \emptyset & \text{if } l \in final(S_*) \\ \bigcup \{ LV_{entry}(l') \mid (l', l) \in flow^R(S_*) \} & \text{otherwise} \end{cases}$$

may

backward

Subject PA
Date ۹۷ ،۱،۲۷

$$LV_{entry}(\ell) = \bigsqcup \left(LV_{exit}(\ell) \setminus kill_{LV}(B^{\ell}) \right) \cup gen_{LV}(B^{\ell})$$

$$B^{\ell} \in Blocks(S_*)$$

Example —

$$(\text{while } [x>1]^{\ell} \text{ do } [skip]^{\ell'}); [x := x+1]^{\ell''}$$

$$LV_{entry}(\ell) = LV_{exit}(\ell) \cup \{x\}$$

$$LV_{entry}(\ell') = LV_{exit}(\ell')$$

$$LV_{entry}(\ell'') = (LV_{exit}(\ell'') \setminus \{x\}) \cup \{x\} = \{x\}$$

$$\downarrow$$

$$\emptyset$$

$$LV_{exit}(\ell) = LV_{entry}(\ell') \cup LV_{entry}(\ell'')$$

$$LV_{exit}(\ell') = LV_{entry}(\ell)$$

$$LV_{exit}(\ell'') = \emptyset$$

می‌شود محاسبه fixed point را با توجه به اینکه ... (نقطه ثابت)

Subject PA
Date ۱۳۹۷/۱/۲۷

برای این روابط تعیین داریم:

$$LV_{exit}(l) = LV_{exit}(l) \cup \{x\}$$

یعنی $LV_{exit}(l)$ ، هر Superset ای از متغیر $\{x\}$ است.

یعنی $\{x\}$ ، $\{x,y\}$ ، ...

مساوی ⟵ smallest ⟵ کمترین

⟵ حقه می‌خواهیم تعیین کنیم کمترین است؟ (زیست‌ها در این اصطلاح مفهوم دارد.)

این ساری که برای کمترین با این مفهوم ، روابط Syntactic (نحوی) هستند با این سوال از جنس Semantic (معنایی)

است.

اصلاح کمترین به روابط Syntactic را در حوزه Static Analysis انجام می‌دهیم.

به شان دهیم که چگونه می‌توان Syntactic کمترین را با مفهوم معنایی Semantics به دست آورد.

(یعنی به روش Semantic-based کمترین به Semantics-directed معروف است.)

هم اجزای روابط Semantics وجود دارد.

برای هر برنامه

پس برای یک برنامه این سوال را با چه معنایی پاسخ می‌دهیم.

P4PCO

۹

Subject
Date

Structural operational semantics

small-step
(evaluation) big-step (SOS)
context

operational
denotational
semantics

axiomatic

fixed point

small-step operational semantics

State:

$\sigma \in State$ $\quad State = Var \to \mathbb{Z}$

Configuration:

1 - a pair of a statement and a state

2 - a state

a state terminal configuration

continuation

Subject PA
Date ۱۳۹۷/۱/۲۷

Transition:

- $\langle S, \sigma \rangle \rightarrow \sigma'$ ... } one step of computation
- $\langle S, \sigma \rangle \rightarrow \langle S', \sigma' \rangle$ ...

... rule ... judgment ...

Interpretation Function:

$$\mathcal{A} : AExp \rightarrow (State \rightarrow \mathbb{Z})$$

↳ a semantic function that gets an arithmetic expression and gives a function state → ℤ

... State ...

abstract syntax tree of the expression

$$\mathcal{A}[\![x]\!]\sigma = \sigma(x)$$

$$\mathcal{A}[\![n]\!]\sigma = \mathcal{N}[\![n]\!]$$

numeral ... numeral

$$\mathcal{A}[\![a_1 \ op_a \ a_2]\!]\sigma = \mathcal{A}[\![a_1]\!]\sigma \ \underline{op_a} \ \mathcal{A}[\![a_2]\!]\sigma$$

... ...

Subject PA

Date ۹۷/۱۲/۱۷

boolean exp

$$\mathcal{B} : BExp \longrightarrow (State \to \mathbb{T})$$

$$\mathcal{B}[\![\text{not } b]\!]\sigma = \neg \mathcal{B}[\![b]\!]\sigma$$

$$\mathcal{B}[\![b_1 \ op_b \ b_2]\!]\sigma = \mathcal{B}[\![b_1]\!]\sigma \ \underline{op_b} \ \mathcal{B}[\![b_2]\!]\sigma$$

$$\mathcal{B}[\![a_1 \ op_r \ a_2]\!]\sigma = \mathcal{A}[\![a_1]\!]\sigma \ \underline{op_r} \ \mathcal{A}[\![a_2]\!]\sigma$$

$$\frac{}{\langle [x := a]^l, \sigma \rangle \to \sigma[x \mapsto \mathcal{A}[\![a]\!]\sigma]}$$

$$\frac{}{\langle [skip]^l, \sigma \rangle \to \sigma}$$

$$\frac{\langle S_1, \sigma \rangle \to \langle S_1', \sigma' \rangle}{\langle S_1 ; S_2, \sigma \rangle \to \langle S_1' ; S_2, \sigma' \rangle}$$

$$\frac{\langle S_1, \sigma \rangle \to \sigma'}{\langle S_1 ; S_2, \sigma \rangle \to \langle S_2, \sigma' \rangle}$$

Subject PA

Date ۱۳۹۷/۱/۲۷

$\langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \rightarrow \langle S_1, \sigma \rangle$ if $\mathcal{B}[\![b]\!]\sigma = true$

$\langle \text{if } [b]^l \text{ then } S_1 \text{ else } S_2, \sigma \rangle \rightarrow \langle S_2, \sigma \rangle$ if $\mathcal{B}[\![b]\!]\sigma = false$

$\langle \text{while } [b]^l \text{ do } S, \sigma \rangle \rightarrow \langle (S; \text{while } [b]^l \text{ do } S), \sigma \rangle$ if $\mathcal{B}[\![b]\!]\sigma = true$

$\langle \text{while } [b]^l \text{ do } S, \sigma \rangle \rightarrow \sigma$ if $\mathcal{B}[\![b]\!]\sigma = false$

Structural Induction ← base rule / proper rule

Lemma –

(i) If $\langle S, \sigma \rangle \rightarrow \sigma'$, then $final(S) = \{fin(S)\}$ (terminating)

Subject
Date

(ii) If $\langle S, \sigma \rangle \longrightarrow \langle S', \sigma' \rangle$, then

$final(S) \supseteq final(S')$,

$flow(S') \supseteq flow(S')$,

$blocks(S') \supseteq blocks(S')$,

and if S is label consistent then so is S'.

(Proof - in text book)

Correctness of Live Variable Analysis:

(correctness ...)

... $LV^{=}(S_*)$ ... statement ...

... $LV^{\subseteq}(S_*)$ ... constraint ...

Superset ...

if $final(S_*)$

$$LV_{exit}^{(l)} \supseteq \begin{cases} \emptyset \\ \cup \{ LV_{entry}(l') \mid (l', l) \in flow^R(S_*) \} \end{cases}$$

$$LV_{entry}(l) \supseteq (LV_{exit}(l) \setminus kill_{LV}(B^l)) \cup gen_{LV}(B^l)$$

Subject PA

Date ۱۳۹۷/۲/۱

A collection $live_{entry}, live_{exit} : Lab_* \to \wp(Var_*)$ of functions

$live$ solves $LV^{=}(S_*)$, written $live \models LV^{=}(S_*)$,

if the functions satisfy the equations.

Similarly for $LV^{\subseteq}(S_*)$.

$(live \models LV^{\subseteq}(S_*))$

Lemma – Consider a label consistent program S_*. If $live \models LV^{=}(S_*)$, then $live \models LV^{\subseteq}(S_*)$. The least solution of $LV^{=}(S_*)$ coincides with the least solution of $LV^{\subseteq}(S_*)$.

Subject

Date ۹۷/۲/۱

$$\begin{cases} live \models LV(s) & \text{iff} \quad live \sqsupseteq F^{s}_{LV}(live) \\ live \models LV^{=}(s) & \text{iff} \quad live = F^{(s)}_{LV}(live) \end{cases}$$

$$lfp(F^{s}_{LV}) = \sqcap \{ live \mid live \sqsupseteq F^{s}_{LV}(live) \}$$

$$= \sqcap \{ live \mid live = F^{s}_{LV}(live) \}$$

$$lfp(F^{s}_{LV}) = F^{s}_{LV}(lfp(F^{s}_{LV})) \quad \text{as well as}$$

$$lfp(F^{s}_{LV}) \sqsupseteq F^{s}_{LV}(lfp(F^{s}_{LV})).$$

Subject PA
Date 1398/12/1

(نیاز label consistent نیز)

Lemma - If $live \models LV^{\subseteq}(S_1)$ and $flow(S_1) \supseteq flow(S_2)$ and $blocks(S_1) \supseteq blocks(S_2)$ then $live \models LV^{\subseteq}(S_2)$.

Corollary - If $live \models LV^{\subseteq}(S)$ and $\langle S, \sigma\rangle \rightarrow \langle S', \sigma'\rangle$ then also $live \models LV^{\subseteq}(S')$.

(Constraint) ها preserve می‌شود. این ویژگی را ⊆ می‌گوییم.

ویژگی (یا property) = صحت‌سازی. پس ویژگی ما invariant property است.

Lemma - If $live \models LV^{\subseteq}(S)$ then for all $(l, l') \in flow(S)$ we have $live(l) \supseteq live(l')$.

exit entry

از ویژگی می‌توان نتیجه گرفت که live در exit بزرگتر از live در entry است.

صحت آنالیز از semantic، correctness نتیجه می‌شود. correctness از ویژگی نتیجه می‌شود.

↓

حال باید نشان دهیم variable های live چه هستند؟

کوچکترین مجموعه‌ای از متغیرها که state را تغییر نمی‌دهد، همان live variable ها هستند.

از ویژگی نتیجه می‌شود که هر مجموعه‌ای که شامل این مجموعه باشد، صحت‌سازی می‌کند.

Subject

Date

بیشتر تعریف‌ها در درس اول از منابع بوده است و تنها تعریف شده است.

در مورد semantics بیشتر توضیح دهید. challenging

Assume that V is a set of live variables and define the

correctness relation:

$$\sigma_1 \sim_V \sigma_2 \text{ iff } \forall x \in V.\ \sigma_1(x) = \sigma_2(x)$$

Example - $[x := y+2]^l$, $V_1 = \{y, z\}$, $V_2 = \{x\}$

$\sigma_1 \sim_{V_1} \sigma_2$ iff $\sigma_1(y) = \sigma_2(y)$ and $\sigma_1(z) = \sigma_2(z)$

$\sigma_1 \sim_{V_2} \sigma_2$ iff $\sigma_1(x) = \sigma_2(x)$

if $\langle [x := y+2]^l , \sigma_1 \rangle \longrightarrow \sigma_1'$ and

$\langle [x := y+2]^l , \sigma_2 \rangle \longrightarrow \sigma_2'$, then

$\sigma_1 \sim_{V_1} \sigma_2$ ensures $\sigma_1' \sim_{V_2} \sigma_2'$.

از V_2 مجموعه متغیرهای زنده پس از اجرای l و V_1 مجموعه متغیرهای زنده پیش از اجرای l.

σ_2 و σ_1 state

---

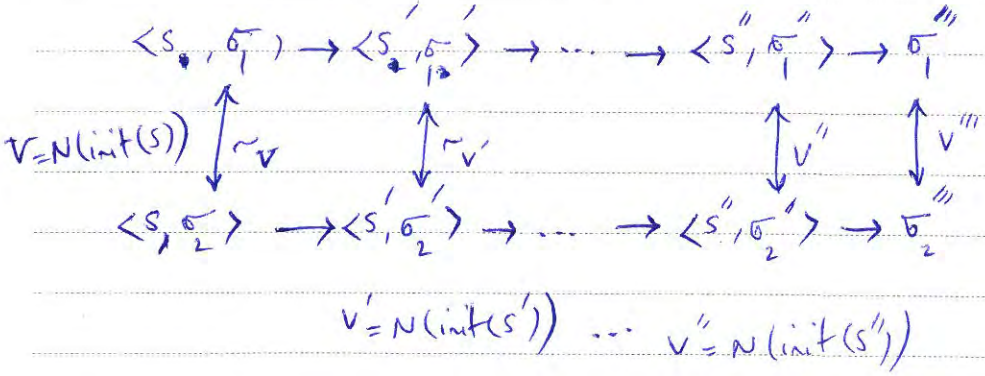
agreement V_1 متغیرهای

Subject PA
Date ۱۳۹۷/۱۲/۱

$N(\ell) = live_{entry}(\ell)$: تعریف می‌شود

$X(\ell) = live_{exit}(\ell)$

Correctness result says that $\sim$ is invariant under the computation.
(Sym relation)



Lemma – Assume $live \models LV^{\subseteq}(s)$. Then

$\sigma_1 \sim_{X(\ell)} \sigma_2$ implies $\sigma_1 \sim_{N(\ell')} \sigma_2$ for all $\ell, \ell' \in flow(s)$.

Theorem – If $live \models LV^{\subseteq}(s)$, then

(i) if $\langle s, \sigma_1 \rangle \to \langle s', \sigma_1' \rangle$ and

$\sigma_1 \sim_{N(init(s))} \sigma_2$, then there exists σ_2' such that

$\langle s, \sigma_2 \rangle \to \langle s', \sigma_2' \rangle$ and $\sigma_1' \sim_{N(init(s'))} \sigma_2'$.

Subject PA
Date ۹۷/۲/۳

(ii) if $\langle S, \sigma_1 \rangle \to \sigma_1'$ and $\sigma_1 \sim_{N(int(S))} \sigma_2$, then there exists σ_2' such that $\langle S, \sigma_2 \rangle \to \sigma_2'$ and

$$\sigma_1' \sim_{X(int(S))} \sigma_2'.$$

(Proof in textbook)

↳ by induction on two judgment forms.

[illegible]

[illegible] Monotone Framework [illegible]

[illegible]

[illegible] scanner, parser [illegible] while language [illegible]

[illegible] data flow analysis [illegible] flow graph [illegible]

[illegible]

[illegible]

PqPCO

Subject PA
Date ۹۷/۱۲/۲

Monotone Frameworks :

$$\text{Analysis}_{\circ}(l) = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup \{ \text{Analysis}_{\bullet}(l') \mid (l', l) \in F \} & \text{otherwise} \end{cases}$$

$$\text{Analysis}_{\bullet}(l) = f_l(\text{Analysis}_{\circ}(l))$$

- $\bigsqcup$ is $\cap$ or $\cup$
- F is either $\text{flow}(S_*)$ or $\text{flow}^R(S_*)$
- E is $\{\text{init}(S_*)\}$ or $\text{final}(S_*)$
- ι specifies the initial or final analysis information
- f_l is the transfer function associated with $B^l \in \text{blocks}(S_*)$

- forward analysis → $F = \text{flow}(S_*)$
- backward analysis → $F = \text{flow}^R(S_*)$

Subject
Date

Analysis • → exit

Analysis ○ → entry

forward تحلیل های

(isolated entries : توجه)

Analysis • → entry

Analysis ○ → exit

backward تحلیل های

(isolated exits : توجه)

$\sqcup$ is $\cap$ → ⊇ superset ⇒ MUST analysis

$\sqcup$ is $\cup$ → ⊆ subset ⇒ MAY analysis

forward, must ⊇ backward, must

forward, may ⊆ backward, may

تحلیل های MUST (ابرمجموعه) دارای بیشترین جواب ممکن هستند greatest

تحلیل های MAY (زیرمجموعه) دارای کمترین جواب ممکن هستند least

		Analysis ○	Analysis •
forward	$F = flow(S_*)$	entry	exit
backward	$F = flow^R(S_*)$	exit	entry

$\sqcup$ is $\cap$	MUST	greatest
$\sqcup$ is $\cup$	MAY	least

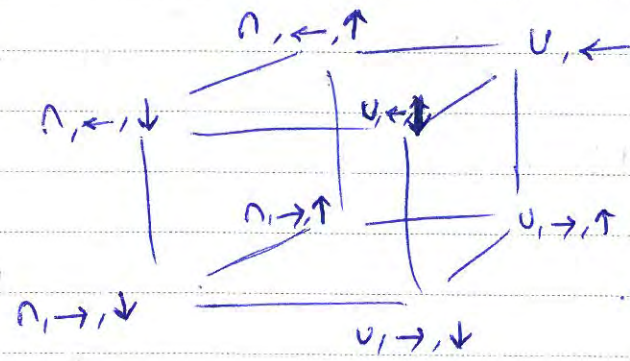
Subject PA

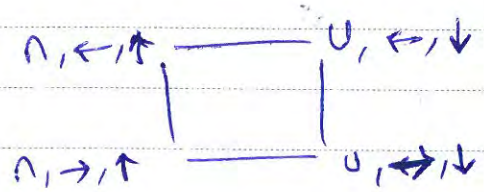
Date ۹۷/۲/۴

$$\{\cap, \cup\} \times \{\rightarrow, \leftarrow\} \times \{\uparrow, \downarrow\}$$

forward backward sfp lfp

(cube)





$$Analysis_0(l) = \bigsqcup \{ Analysis_\bullet(l') \mid (l', l) \in F \} \sqcup \iota_E^l$$

$$\text{where } \iota_E^l = \begin{cases} \iota & \text{if } l \in E \\ \bot & \text{if } l \notin E \end{cases}$$

$$Analysis_\bullet(l) = f_l(Analysis_0(l))$$

PqPCO

Subject
Date

property space ... framework ...

Property Space (L) ⟶ فضای اطلاعات مورد نظر ما

Combination Operator $\sqcup : \wp(L) \to L$

(مجموعه‌ای از property ها) set ... ورودی

$\sqcup : L \times L \to L$ (دو تایی)

$\sqcup\{l_1, l_2\} = l_1 \sqcup l_2$

$\sqcup \emptyset = \bot$

$(L, \sqsubseteq)$ a poset

(superset ، subset) ... رابطه ترتیبی

Complete lattice ← وجود sup برای هر زیرمجموعه

(یا حداقل semi-lattice ، وجود sup برای هر دو عضو)

$(l_n)_n$, $l_1 \sqsubseteq l_2 \sqsubseteq \cdots$ ... ascending chain condition

eventually stabilizes $\exists n.\ l_n = l_{n+1} = \cdots$

Subject PA
Date ۹۷/۴/۳

$l_1 \sqsubseteq l_2$ iff $l_1 \sqcup l_2 = l_2$

$\sqsubseteq$ precedence

$\sqcup$ lub

Subset ، precedence

Example -

Reaching Definition Analysis

$L = \mathcal{P}(Var_* \times Lab_*^?)$

transfer

iteration

$\sqsubseteq = \subseteq$

$\sqcup = \cup$

$\bot = \emptyset$

Ascending chain condition → $(Var_* \times Lab_*^?)$

(ACC)

finite

Subject
Date

(این جلسه تمرکز بر intra-procedural و تحلیل‌های این دسته است ← مسیر زیرین)

Available Expressions Analysis:

$$L = \wp(AExp^*_*)$$

$$\sqsubseteq = \supseteq$$

$$\sqcup y = \cap y$$

$$\bot = AExp_*$$

زیرا ACC ← تمام AExp$_*$

در این جلسه، تمرکز بر transfer function‌ها:

$$f_l : L \to L$$

property ←

$$l \sqsubseteq l' \Rightarrow f_l(x) \sqsubseteq f_{l'}(x)$$

monotone یعنی

ارائه‌ی روشی برای اثبات این خاصیت

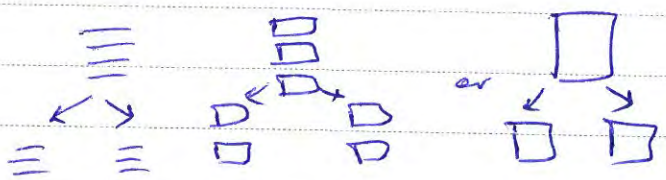
اثبات درستی الگوریتم

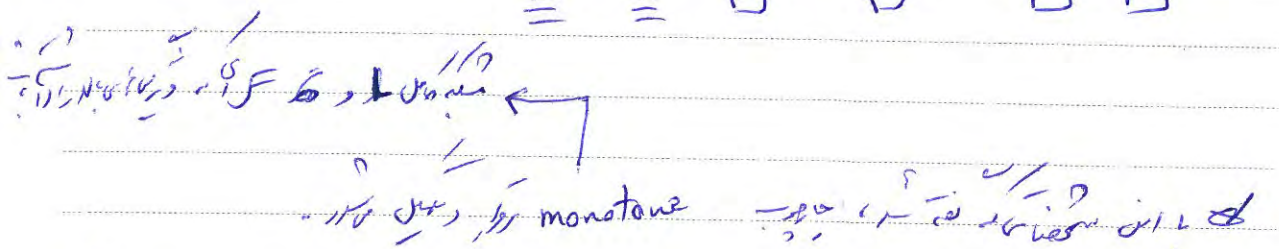
Monotone Framework در این درس F

PqPCO

Subject PA
Date ۹۷/۲/۳

- contains all transfer functions f_ℓ in question
- contains the identity function id
- closed under composition of functions (Sequencing)





- Complete lattice L
- transfer functions F
- flows F
- External Labels E
- Extremal Values $\in L$

• bullet
○ circle
• dot

Subject
Date

(f_dot)

- mapping $f_\bullet : Lab_\star \to \mathcal{F}$

$(L, \mathcal{F}, F, E, \iota, f_\bullet)$

Analysis$^{=}$, Analysis$^{\equiv}$

Constraint-based equational

Figure 2.6

Lemma - Every analysis given in Figure 2.6 is an instance of the Monotone Framework.

Subject PA
Date ۱۳۹۷/۲/۸

Project ← برنامه دوره ای برای LNW برای بررسی برنامه برای کاربرد
نتایج کلی استفاده از این روش را استخراج کرده، طی این روش را بررسی کنید.

* کلیه های این روش distributive هستند، یعنی نتایج transfer برای هر block. البته این
روش distributive را در framework monotone بررسی کنیم، چون این روش را می توان برای
از این روش.

* با توجه به framework های این روش ها را برای این جا بررسی کنیم و برای هر کدام از این جا بررسی
بسازیم. این روش ها generic هستند. از این روش ها می توان برای بررسی کلیه کاربردها

The MFP solution:

(Maximal Fixed Point)

این روش greatest و least fixed point را بررسی کرده.
نحوه این روش، توسط Chaotic Iteration بررسی می شود.

Analysis این → اطلاعات Analysis برای کدام قسمت
از برنامه باید بررسی شود.

Worklist W → لیستی (مجموعه) از pair ها
a list of pairs of F
↓
2 flow

PqPCO The presence of a pair in the worklist indicates that
the analysis has changed at exit of the block labelled

109

Subject
Date

by the first component and so must be recomputed at the entry to the block labelled by the second component.

INPUT : An instance of the monotone framework $(L, \sqcup, F, E, \iota, f_\cdot)$

OUTPUT : $MFP_\circ$, $MFP_\bullet$

METHOD : step 1 : Initialization

$W := nil$

for all $(l, l') \in F$ do $W := cons((l, l'), W)$;

for all l in F or E do

if $l \in E$ then $Analysis[l] := \iota$;

else $Analysis[l] := \bot_L$;

Subject PA
Date 1397, 1, 18

step 2 : Iteration

while $W \neq nil$ do

$l := First(head(W))$;
$l' = second(head(W))$;

$W := tail(W)$;

if $f_l(Analysis[l]) \not\sqsubseteq Analysis[l']$ then

$Analysis[l'] := Analysis[l'] \sqcup f_l(Analysis[l])$;

for all l'' with $(l', l'') : F$ do

$W := Cons((l', l''), W)$;

Step 3 : Result

for all $l : F$ or E do

$MFP_o(l) := Analysis[l]$;

$MFP_\bullet(l) := f_l(Analysis[l])$;

PAPCO

iterative نه:

Subject

Date

Example – $[x := a+b]^1 ; [y := a*b]^2 ; \text{while } [y > a+b]^3 \text{ do}$

$([a := a+1]^4 ; [x := a+b]^5)$;

- $a+b$: Available Expression

↓

- lattice $\mathcal{P}(AExp_*)$

Available Expressions:

- $\sqsubseteq = \supseteq$
- MUST analysis $\rightarrow \sqcup = \cap$
- $\perp = AExp_*$
- $\iota = \emptyset$

$E = init(S_*)$

$F = F_{lab}(S_*)$

$U = \{a+b, a*b, a+1\}$

$W = ((2,3), (3,4), (4,5), (5,3))$

Analysis[l]:

step	W	1	2	3	4	5
1	$((1,2), W)$	$\emptyset$	$U = AExp_*$	U	U	U
2	$((2,3), W)$	$\emptyset$	$\{a+b\}$	U	U	U
3	$((3,4), W)$	$\emptyset$	$\{a+b\}$	$\{a+b, a*b\}$	U	U
4	$((4,5), W)$	$\emptyset$	$\{a+b\}$	$\{a+b, a*b\}$	$\{a+b, a*b\}$	U
5	$((5,3), W)$	$\emptyset$	$\{a+b\}$	$\{a+b, a*b\}$	$\{a+b, a*b\}$	$\emptyset$
6	$((3,4), W)$	$\emptyset$	$\{a+b\}$	$\{a+b\}$	$\{a+b\}$	$\emptyset$

Subject PA
Date ۱۳۹۷/۲/۸

هر کدام از ستون‌ها برای یک label است.

	W	1	2	3	4	5
11	((5,3))	∅	{a+b}	{a+b}	{a+b}	∅
12	()	∅	{a+b}	{a+b}	{a+b}	∅

$MFP_0(\ell)$ = ستون ℓ در سطر 12

$$\begin{cases} MFP_0(1),\ MFP_0(5) = \emptyset\ \emptyset \\ MFP_0(i) = \{a+b\} \quad i=2,3,4 \end{cases}$$

این راه حل، کوچکترین fixed point است.

Semantics: معنا ← "درست"! Correct

آیا این الگوریتم کار درستی انجام می‌دهد؟ ← معنای برنامه

Lemma - The worklist algorithm always terminates and it computes the least solution to the instance of the framework given as input.

Subject PA

Date 97.2.8

Proof - (in text book)

termination → step 1 and 3 → bounded

step 2 و تعداد b ثابت است. پس تعداد دفعات اجرای حلقه

worklist اول و b^2 عدد pair خواهد

هر بار یک عنصر از worklist حذف می‌شود و یک عنصر اضافه می‌شود.

اگر اضافه نشود → تعداد عناصر worklist در هر مرحله کاهش می‌یابد.

تعداد دفعات اجرای حلقه (while) محدود است.

تعداد دفعات اجرای حلقه ACC را نداریم ⟸ تعداد عناصر worklist ثابت می‌ماند.

⟸ متناهی است.

correctness →

Analysis ، Analysis

Analysis توسط

(i) و

Analysis ⟸ proceed

Subject PA
Date

(ii) Analysis$_0$ is an approximation of Analysis*

(iii) combining the result

از هر دو بیت از هم جدا شده، بیت متناظر (که قبلا بیت متناظر precede می‌کرد)

$\forall l$: Analysis$_0$ [l] $\sqsubseteq$ Analysis* (l)

Analysis$_0$ [l] $\sqsubseteq$ Analysis* for all l : عبارت قبل را به این صورت می‌توان بیان کرد

After step 1 ← بعد از اجرای گام اول

گام دوم: بیت از هم جدا شده: هیچ تغییری نمی‌کند

Analysis [l''] is unchanged for all l'' except for some l'.

برای یال‌های تغییر یافته:

$(l, l') \in F$

new Analysis [l'] = old Analysis [l'] $\sqcup$ f_l (old Analysis [l])

$\sqsubseteq$ Analysis$_0$ (l') $\sqcup$ f_l (Analysis$_0$ [l]) (چون monotone است)

= Analysis$_0$ (l')

پس ثابت شد. ← این یکی از خصوصیات framework است

Subject
Date

MFP ... least fixed point ...

worst case:

$O(b^2 \cdot h)$ ... complexity ...

b = number of labels of the program

h = height of the lattice (finite height)

$h \leq v \cdot b$ ... Reaching Definitions ...

$O(v \cdot b^3)$

$P(Var_* \times Lab_*^?)$

... iterative ...

[MOP] ... meet over all paths

project

transfer ... MFP ...

... correctness ... MFP ... distributive

Subject:
Year. Month. Date. ۹۷/۲/۱۵

~~Ready~~ ~~Stuff~~ به کدام نمونه در

Reaching Definition.

$h \leq (V, b)$ ~~$\beta \neq 38$~~

$P(War_x \times Lab_x^?)$ $O(v, b^3)$

MOP (Meet over all Paths)

- این گذر جهش به جای meet با join والی می‌رود.

join (⊔) (⊑) least upper bound greatest lower bound ← meet

- هر خواص اطلاعاتی را در نظر می‌گیریم که بعد از ترکیب دو مسیر آن را هم داشته باشد MOP

- MFP عملاً جواب را برمی‌گرداند اما MOP ممکن است در loop ها نهایت بیاید.

- حال یک instance از Monotone Framework در نظر می‌گیریم:

$(L, \wedge, F, E, \iota, f.)$

- برای یک مکان از label ها در نظر می‌گیریم $\vec{l} = [l_1, \dots, l_n]$

$n \geq 0$, a sequence of labels

if $n=0$ → یعنی label ها $\emptyset$ است.

- دو مجموعه مسیر را توسط notation بر اساس این تعریف کرد.

- چیزی باشد که approximate شود؟ ما که داریم عملکرد را دقیق انجام می‌دهیم پس چرا approximate می‌گیریم؟

Subject:
Year. Month. Date. ( )

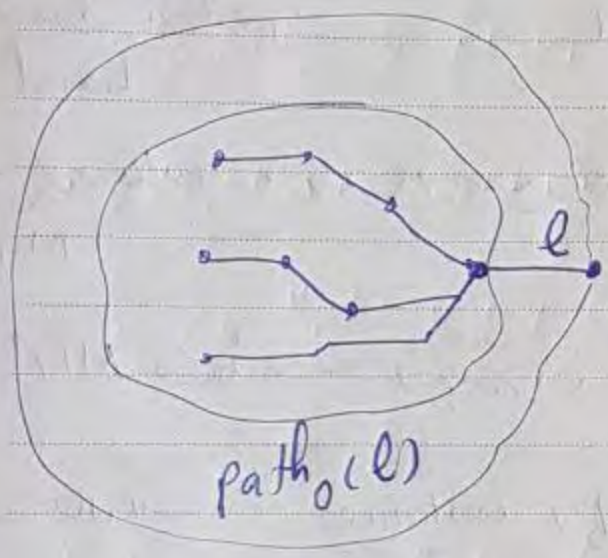
$$Path_0(l) = \{ [l_1, \dots l_n] \mid n \geq 1 \wedge \forall i < n \,.\, (l_i, l_{i+1}) \in F \wedge l_n = l \wedge l_1 \in E \}$$

مجموعه مسیرهایی تا l، البته خود
l داخل آن نمی‌شود (روی فلوگراف)

اگر forward است یعنی از start برنامه تا قبل از l
اگر backward است یعنی از end برنامه تا قبل از l

$$Path_0(l) = \{ [l_1, \dots l_n] \mid n \geq 1 \wedge \forall i < n \,.\, (l_i, l_{i+1}) \in F \wedge l_n = l \wedge l_1 \in E \}$$

مجموعه مسیرهایی که از شامل خود l ←



- این که دارد که مجموعه مسیرهایی که می‌بینید برای کار approximate است؟
چون این مسیرها Synthactic حساب می‌شوند نه Semantically!
بنابراین ممکن است بعضی از مسیرها اصلاً عملی نشوند! مثلاً

```
if (halting problem)
    then
        :
    end
```

flow graph, parse tree ممکن است نباشد اما در این مسیرها در semantic خطا هستند.

Subject:
Year. Month. Date. ( )

For $\vec{l} = [l_1, \dots, l_n]$, we define the transfer function

$$f_{\vec{l}} = f_{l_n} \circ \dots f_{l_1} \circ (id) \leftarrow \text{تابع همانی}$$

این تعاریف که بدست آوردیم از ازای هر یک transfer function بروی یک label یک مسیر است. اگر n=0 باشد باید از ابتدا یک چیزی داشته باشیم (شروع برنامه) و این را با id نشان می‌دهیم.

حال $MoP_\circ$ ، $MoP_\bullet$ درست می‌کنیم.

bullet circle

iota

$$MoP_\circ(l) = \bigsqcup \{ f_{\vec{l}}(\iota) \mid \vec{l} \in Path_\circ(l) \}$$

روی همه مسیرهای تعاریف را با هم تجمیع کنیم.

$$MoP_\circ(l) = \bigsqcup \{ f_{\vec{l}}(\iota) \mid \vec{l} \in Path_\circ(l) \}$$

MOP

از روی الگوریتم پایان پذیر است! برعکس MFP که با افزایش Ascending همیشه پایان پذیر است.

- بین MFP و MOP چه رابطه‌ای برقرار است؟

Lemma: Consider the MFP and MOP solutions to an instance $(L, F, \sqcap, E, \iota, f_\circ)$ of a monotone framework; then

$$MFP_\circ \sqsupseteq MoP_\circ \quad \text{and} \quad MFP_\bullet \sqsupseteq MoP_\bullet$$

If the framework is distributive and $path_\circ(l) \neq \emptyset$ for all l in E and F, then

$$MFP_\circ = MoP_\circ \quad \text{and} \quad MFP_\bullet = MoP_\bullet$$

Subject:
Year. Month. Date. ( )

MOP یک approximation از MFP است.

- کتاب Gold Reich را در مورد Complexity بخوانید.
- چرا که بعید است NP ≠ P ؟
چرا که بعید است در دنیا داشته باشیم که هر مسئله‌ای، چک کردن جواب آن مسئله راحت تر نباشد!

- در جایی که اشتراک داریم (must analysis)، ∃ می‌شود ∀ !

$$MFP_\circ \sqsubseteq MOP_\circ$$

- در MOP، MFP، یکی کل اطلاعات را جمع می‌کند و بعد تابع را روی آن اعمال می‌کند در حالیکه در دیگری تابع را روی اطلاعات موجود در طول مسیر اعمال می‌کنم و در نهایت آنها را جمع می‌کنم

و شرط distributivity برقرار باشد ((lemma) قبل)، این ۲ تا برابر می‌شوند.

اثبات lemma:

$$\forall \ell .\ MOP_\circ(\ell) \sqsubseteq f_\ell(MOP_\circ(\ell))$$

$$\forall \ell .\ MFP_\circ(\ell) = f_\ell(MFP_\circ(\ell))$$

$MFP_\circ \sqsupseteq MOP_\circ$ کافیست $MFP_\circ \sqsubseteq MOP_\circ$ را اثبات کنیم (و رابطه برعکس)

خودکفا بنابر روابط بالا خودکفا هم برآورده می‌شود

$$\forall \ell \quad MOP_\circ(\ell) \sqsubseteq MFP_\circ(\ell)$$

$MFP_\circ$ is the least fixed point of the functional G defined by

$$G(A_\circ)(\ell) = \left(\bigsqcup \{ f_{\ell'}(A_\circ(\ell')) \mid (\ell,\ell') \in F \} \right) \sqcup \iota^{\ell}_{E}$$

F مربوط به نمونه

PAPCO

Subject:
Year. Month. Date. ( )

حالا تعریف کنیم برای مسیرهایی با طول n است:

Define

$$MoP_o^n(\ell) = \bigsqcup \{ f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in Path_o(\ell), |\vec{\ell}| < n \}$$

$$MoP_o(\ell) = \bigsqcup_n MoP_o^n(\ell)$$

join همه مسیرها

To prove $MoP_o \sqsubseteq MFP_o$ is therefore suffices to prove

$$\forall n.\ MoP_o \sqsubseteq MFP_o$$

کافی است این را ثابت کنیم.
این کار را با استقرا انجام می‌دهیم:

Induction: $MoP_o^0 \sqsubseteq MFP_o$ (Basis)

این صفر می‌شود ← $\bot$

Step

$$MFP_o(\ell) = G(MFP_o)(\ell)$$

این از تعریف G آمده

$$= \left(\bigsqcup \{ f_{\ell'}(MFP_o(\ell')) \mid (\ell', \ell) \in F \} \right) \sqcup \iota_E^\ell$$

$$\sqsupseteq \left(\bigsqcup \{ f_{\ell'}(MoP_o^n(\ell')) \mid (\ell', \ell) \in F \} \right) \sqcup \iota_E^\ell$$

(Induction Hypothesis) یعنی از بزرگتر است. فرض کنیم برای n

ساده‌سازی اطراف بالا

$$= \left(\bigsqcup \{ f_{\ell'}(\bigsqcup \{ f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in Path_o(\ell'), |\vec{\ell}| < n \}) \mid (\ell', \ell) \in F \} \right) \sqcup \iota_E^\ell$$

$$\sqsupseteq \left(\bigsqcup \{ \ell \sqcup f_{\ell'}(f_{\vec{\ell}}(\iota)) \mid \vec{\ell} \in Path_o(\ell'), |\vec{\ell}| < n \} \right.$$

$$\left. \mid (\ell', \ell) \in F \right) \sqcup \iota_E^\ell$$

$$= \bigsqcup \{ f_{\vec{l}}(\iota) \mid \vec{l} \in Path_0(l),\ 1 \le |\vec{l}| \le n \}) \sqcup \iota_E^l$$

$$= MOP_0^{n+1}$$

می‌ثابت شد که:

$$MOP_0(l) \sqsubseteq MFP_0(l)$$

در حالتی که distributive باشد (در این صورت مثل بالای) نیز ثابت می‌شود و این ۲ برابر می‌شوند.

مثالی برای intra procedural کار کردیم. اگر بخواهیم inter procedural را بررسی کنیم باید call graph را نیز در نظر بگیریم.

Subject PA

Date ۹۷/۲/۱۷

Interprocedural Analysis

تا این جزء از Function ، Call ، procedure ، ... بود که ... را بررسی کنیم.

scope ، activation records ، environment ، ... procedure ... parameter Passing

... Syntax ...

P_*

begin D_* S_* end : ... ساختار کلی برنامه

$D_* ::= \text{Proc}\ p(\text{val}\ x, \text{res}\ y)\ \text{is}^{l_n}\ S\ \text{end}^{l_x} \mid DD$

Pass-by-value ، Pass-by-result ، delimiter

$S ::= \ldots \mid [\text{call}\ p(a,z)]^{l_c}_{l_r}$

$l_c \rightarrow$ call label

$l_r \rightarrow$ return label

Subject

Date 97.2.17

$(v = fib(z) + u)$

Example –

begin proc fib (val z, u, res v) is^1

if $[z<3]^2$ then $[v := u+1]^3$

else $([call\ fib(z-1, u, v)]^4_5$;

$[call\ fib(z-2, v, v)]^6_7$)

end^8 ;

$[call\ fib(x, 0, y)]^9_{10}$

end

Flow Graph for statements:

: {statement ... ...

$init([call\ p(a,2)]^{l_c}_{l_r}) = l_c$

$final([call\ p(a,2)]^{l_c}_{l_r}) = \{l_r\}$

$blocks(\quad) = \{[call\ p(a,2)]^{l_c}_{l_r}\}$

$labels(\quad) = \{l_c, l_r\}$

$flow(\quad) = \{(l_c, l_n), (l_x, l_r)\}$

if proc p (val x, res y) is l_n s end l_x is in D_*

Subject PA
Date ۱۴۰۲/۱۲/۱۷

(؟ ...) flow ... ...

... unique ... ...

... dynamic ... ...
dispatching

Flow Graph for programs

$init(p) = l_n$

↙

a procedure

$final(p) = \{l_x\}$

$blocks(p) = \{ is^{l_n}, end^{l_x} \} \cup blocks(S)$

$labels(p) = \{l_n, l_x\} \cup labels(S)$

$flow(p) = \{(l_n, init(S))\} \cup flow(S) \cup \{(l, l_x) \mid l \in final(S)\}$

P_* ...

$init_* = init(S_*)$

↳ ... main ...

$final_* = final(S_*)$

Subject
Date

$$blocks_* = \bigcup \{ blocks(p) \mid proc\ p(val\ x, res\ y)\ is^{l_n}\ S\ end^{l_n}\ is\ D_* \} \cup blocks(S_*)$$

$$Lab_* = labels_* = \bigcup \{ labels(p) \mid \dots \} \cup labels(S_*)$$

$$flow_* = \bigcup \{ flow(p) \mid \dots \} \cup flow(S_*)$$

$$inter\_flow_* = \{ (l_c, l_n, l_x, l_r) \mid P_* \text{ contains } [call\ p(a,2)]^{l_c}_{l_r} \text{ as well as } proc\ p(val\ x, res\ y)\ is^{l_n}\ S\ end^{l_x} \}$$

Example -

$$flow_* = \{(1,2),(2,3),(3,8),(2,4),(4,1),(8,5),(5,6),(6,1),(8,7),(7,8),(9,1),(8,10)\}$$

$$inter\_flow_* = \{(9,1,8,10),(4,1,8,5),(6,1,8,7)\}$$

(incarnation: تجسد)

Subject PA
Date ۱۳۹۷/۲/۱۷

$init_* = 9$

$final_* = \{10\}$

backward, forward این‌ها برای interprocedural هستند

Forward : $F = flow_*$, $E = \{init_*\}$, $IF = inter\_flow_*$

Backward : $F = flow_*^R$, $E = final_*$, $IF = inter\_flow_*^R$

structural operational semantics: برای syntax است، semantics برای معنی است

environment: برای global, local است

variable incarnation: تجسد متغیر

An infinite set of locations

$\xi \in Loc$ /xsi/

An environment ρ will map the variables in the current scope to their locations.

$\rho \in Env = Var_* \to Loc$

Subject
Date

and a store σ will then specify the values of these locations

$$\sigma \in Store = Loc \xrightarrow[fin]{} \mathbb{Z}$$

$$\sigma \in State : Var^* \to \mathbb{Z}$$

$$\sigma \circ \rho : Var^* \to \mathbb{Z}$$

$$Loc \to \mathbb{Z} \qquad Var^* \to Loc$$

global environment: ρ_*

$$\rho \vdash_X \langle S, \sigma \rangle \to \langle s', \sigma' \rangle$$

Subject PA
Date ۹۰۲۰۲۲

$$\rho \vdash_* \langle S, \sigma \rangle \xrightarrow{\ell} \sigma' \qquad \rho \vdash_* \langle S, \sigma \rangle \to \sigma' \quad S'$$

$$\rho \vdash_* \langle x := a, \sigma \rangle \to \sigma[\rho(x) \mapsto \mathcal{A}[\![a]\!](\sigma \circ \rho)]$$

(arith exp)

if $\sigma \circ \rho$ is total

$$\rho \vdash_* \langle [\text{call } p(a,z)]^{\ell_c}_{\ell_r}, \sigma \rangle \to \langle \text{bind } \rho_* [x \mapsto \xi_1, y \mapsto \xi_2] \text{ in } S \text{ then } z := y,$$

$$\sigma[\xi_1 \mapsto \mathcal{A}[\![a]\!](\sigma \circ \rho), \quad \xi_2 \mapsto v] \rangle$$

where $\xi_1, \xi_2 \notin \text{dom}(\sigma)$, $v \in \mathbb{Z}$, and

proc p(Val x, res y) is S and ℓ_x is in D_*

$$\frac{\rho' \vdash_* \langle S, \sigma \rangle \to \langle S', \sigma' \rangle}{\rho \vdash_* \langle \text{bind } \rho' \text{ in } S \text{ then } z := y, \sigma \rangle \to \langle \text{bind } \rho' \text{ in } S' \text{ then } z := y, \sigma' \rangle}$$

$$\frac{\rho' \vdash_* \langle S, \sigma \rangle \to \sigma'}{\rho \vdash_* \langle \text{bind } \rho' \text{ in } S \text{ then } z := y, \sigma \rangle \to \sigma'[\rho(z) \to \sigma'(\rho'(y))]}$$

Subject
Date

مثال: برنامه‌ساده‌ای بنویسید و دستگاهی را طوری بدهید را بررسی کنید.

begin proc p (val x , res y) is^{1}

if [x < 1]^{2} then [y := x+1]^{3}

else [call p(x-1, y)]^{4,5} end^{6};

[call p(a,2)]$^{7}_{8}$;

end^{9};

با مقادیر اولیه x=1 برنامه چه کار انجام می‌دهد.

* برای برقراری ارتباط بین توابع: inter-procedural

اولین ایده این است که همان intra-proc را توسعه دهیم به inter-proc

تعریف کنیم.

[call p(a,2)]$^{l_c}_{l_r}$

f_{l_r} , f_{l_c} transfer توابع

proc p(val x, res y) is^{l_n} S end$^{l_{nx}}$: f_{l_n} , $f_{l_{nx}}$

برای برقراری خواص property می‌توان از تحلیل‌های مختلفی استفاده کرد.

Subject PA
Date ۱۳۹۷, ۲, ۲۲

monotone framework

$$(L, \sqcap, F, E, \iota, f_\cdot)$$

flow

(F)

(l_1, l_2) , (l_3, l_4), (l_5, l_6)

proc, call

identity

transfer

f_{l_3} , f_{l_5}

$$A_\cdot(l) = f_l(A_0(l))$$

$$A_0(l) = \bigsqcap \{ A_\cdot(l') \mid (l', l) \in F \text{ or } (l'; l) \in F \} \sqcup \iota_E^l$$

$$\iota_E^l = \begin{cases} \iota & \text{if } l \in E \\ \bot & \text{if } l \notin E \end{cases}$$

Subject
Date

(ادامه جلسه قبل)

inter-proc ، inter-proc ، ...

transfer ، identity

join

(imprecise)

Safe

return

call

inter-flow

(Amir Rnaldi)

Context-Insensitive Analysis

Context-sensitive

MVP (Map ...)

meet over all valid paths

(عملیات meet روی همه مسیرهای معتبر انجام شود)

Subject PA
Date ۱۳۹۷/۲/۲۴

Valid Paths :

P_* begin D_* S_* end
(program)

- Complete path :

$$cp_{l_1,l_2} \longrightarrow l_1 \quad \text{if } l_1 = l_2$$

$$cp_{l_1,l_3} \longrightarrow l_1, cp_{l_2,l_3} \quad \text{if } (l_1,l_2) \in F$$

$$cp_{l_c,l} \longrightarrow l_c, cp_{l_n,l_x}, cp_{l_r,l} \quad \text{if } (l_c,l_n,l_x,l_r) \in IF$$

(body) (return)

$$cp_{9,10} \rightarrow 9,1,cp_{2,8},cp_{10,10}$$

$$cp \rightarrow 9,1,2,cp_{3,8},cp_{10,10} \rightarrow 9,1,2,3,cp_{8,8},cp_{10,10}$$

$$\rightarrow 9,1,2,3,8,10$$

Subject: — ? ?

Date:

A path is said to be a valid path if it starts at an extremal label node of P_* and if all the procedure exits, match the procedure entries, but it is possible that some procedures are entered but not yet exited.

$VP_* \rightarrow VP_{l_1,l_2}$ if $l_1 \in E$, $l_2 \in Lab_*$

$VP_{l_1,l_2} \rightarrow l_1$ if $l_1 = l_2$

$VP_{l_1,l_3} \rightarrow l_1, VP_{l_2,l_3}$ if $(l_1,l_2) \in F$

$VP_{l_c,l} \rightarrow l_c, CP_{l_n,l_x}, VP_{l_r,l}$ if $(l_c,l_n,l_x,l_r) \in IF$

$VP_{l_c,l} \rightarrow l_c, VP_{l_n,l}$ if $(l_c,l_n,l_x,l_r) \in IF$

? ... ? return ...

Subject PA
Date ۱۴۰۲/۲/۲۴

$$VPath_0(\ell) = \{ [\ell_1, \ldots, \ell_{n-1}] \mid n \geq 1 \wedge \ell_n = \ell \wedge [\ell_1, \ldots, \ell_n] \text{ is a valid path} \}$$

$$VPath_\bullet(\ell) = \{ [\ell_1, \ldots, \ell_n] \mid n \geq 1 \wedge \ell_n = \ell \wedge [\ell_1, \ldots, \ell_n] \text{ valid path} \}$$

$$MVP_0(\ell) = \bigsqcup \{ f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in VPath_0(\ell) \}$$

ارتفاع شبکه transfer این

$$MVP_\bullet(\ell) = \bigsqcup \{ f_{\vec{\ell}}(\iota) \mid \vec{\ell} \in VPath_\bullet(\ell) \}$$

enumerate

MOP

$$MVP_0(\ell) \sqsubseteq MOP_0(\ell) \quad , \quad MVP_\bullet(\ell) \sqsubseteq MOP_\bullet(\ell)$$

$VPath \subseteq Paths$

inter-proc

Subject

Date ۱۳۹۷/۲/۲۴

Subject PA
Date ۲۰۱۸/۱۲/۲۴

explicit ... Context ... MFP ...

... Context ...

$\delta \in \Delta$ Context Information

... encoding ... Context ...

... Context ... intra-proc ...

... the intra-procedural fragment

... monotone framework ... instance ...

$(L, \sqcap, F, E, \iota, f_\cdot)$

lattice of properties

$(\hat{L}, \hat{\sqcap}, F, E, \hat{\iota}, \hat{f}_\cdot)$ embellished monotone framework

- $\hat{L} = \Delta \to L$

... Property ... Context Info ... $\hat{L}$ ...

... Context ...

Subject
Date

- $\hat{F}$ consists of monotone functions

- each transfer function $\hat{f}_\ell$ is given by

$$\hat{f}_\ell : (\Delta \to L) \to (\Delta \to L)$$

$$\hat{f}_\ell(\hat{\iota})(\delta) = \hat{f}_\ell(\hat{L}(\delta))$$

$$A_\circ(\ell) = \hat{f}_\ell(A_\circ(\ell))$$

$A_\circ, A_\circ \in \hat{L}$

for all labels that do not label a procedure call
(not ℓ_c or ℓ_r)

$$A_\circ(\ell) = \bigsqcup \{ A_\circ(\ell') \mid (\ell', \ell) \in F \text{ or } (\ell'; \ell) \in F \} \sqcup \hat{\iota}^{\ell}_{E}$$

for all labels

Example - $(L_{Sign}, F_{Sign}, F, E, \ell_{Sign}, f^{Sign}_{\cdot})$

Detection of sign analysis

$$L_{Sign} = \mathcal{P}(Var_* \to Sign) \qquad Sign = \{0, -, +\}$$

(abstract state set)

Subject PA
Date ۱۳۹۷/۴/۲۵

$[x := a]^\ell$

$$f_\ell^{Sign}(Y) = \bigcup \left\{ \phi_\ell^{Sign}(\sigma^{Sign}) \;\middle|\; \sigma^{Sign} \in Y \right\}$$

abstract states مجموعه ← $Y \subseteq Var_\star \to Sign$

$$\phi_\ell^{Sign}(\sigma^{Sign}) = \left\{ \sigma^{Sign}[x \mapsto s] \;\middle|\; s \in A_{Sign}[\![a]\!](\sigma^{Sign}) \right\}$$

$$A_{Sign} : AExp \to (Var_\star \to Sign) \to \mathcal{P}(Sign)$$

$\hat{L}_{Sign} = \Delta \to L_{Sign}$

$\hat{L}_{Sign} = \mathcal{P}(\Delta \times (Var_\star \to Sign))$

$[x := a]^\ell$

$$f_\ell^{Sign}(Z) = \bigcup \left\{ \{\delta\} \times \phi_\ell^{Sign}(\sigma^{Sign}) \;\middle|\; (\delta, \sigma^{Sign}) \in Z \right\}$$

$Z \in \mathcal{P}(\Delta \times (Var_\star \to Sign))$

$\Delta \to \mathcal{P}(D)$ isomorphic : اثبات کنید $\mathcal{P}(\Delta \times D)$

PqPCO

۱۴۴/

Subject PA
Date ۹۷/۲/۲۹

proc p(val x, res y) is l_n S end l_x

$$\hat{f}_{l_n}, \hat{f}_{l_x} : (\Delta \to L) \to (\Delta \to L)$$

$$\hat{f}_{l_n}(\hat{l}) = \hat{l}, \quad \hat{f}_{l_x}(\hat{l}) = \hat{l} \qquad \hat{l} \in \hat{L}$$

identity ← توابع

Context ، این یک ... procedure ... توابع ... (این ... توابع)

... Property ... این ... این ... این ...

Context ... transfer ... l_n, l_x ... این ... Context ...

... (call site) ... Context ...

... این ... این ...

... transfer ... این ... این ...

... property ... procedure ... transfer ... l_x ... این ...

... procedure ... این ...

PAPCO

(stack)

Subject PA
Date ۱۳۹۷/۲/۲۹

$$\left[\text{call } p(a,2)\right]_{l_r}^{l_c}$$

$$\hat{f}^1_{l_c} : (\Delta \to L) \to (\Delta \to L)$$

$$A_0(l_c) = \hat{f}^1_{l_c}\left(A_0(l_c)\right) \quad \text{for all } (l_c, l_n, l_x, l_r) \in \text{IF}$$

$$\hat{f}^2_{l_c, l_r} : (\Delta \to L) \times (\Delta \to L) \to \Delta \to L$$

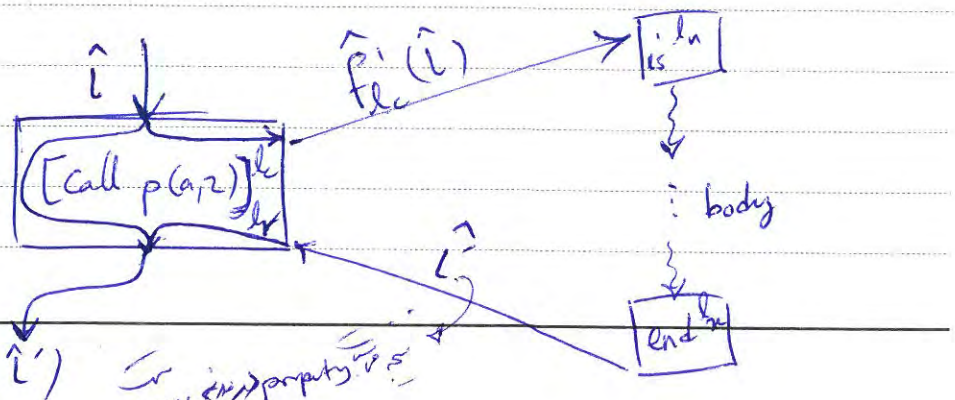
property , Context

property , Context

body

return property , Context

$$A_0(l_r) = \hat{f}^2_{l_c, l_r}\left(A_0(l_c), A_0(l_r)\right)$$



$f^2_{l_c, l_r}(\hat{i}, \hat{i}')$

Subject

Date ۹۷/۲/۲۹

Subject PA
Date ۱۳۹۷,۲,۲۹

- اولین آنالیز پویا که نسبتاً call strings of unbounded length context

اطلاعات مسیرهای اجرا را در context نگه می‌دارد و اطلاعات مسیر را

(مسیر فراخوانی‌ها) call ها ذخیره می‌کند.

↳ یک لیست از برچسب‌های فراخوانی‌ها

$(l_c; l_n)$

- $\Delta = Lab^*$

the most recent label l_c of a procedure call is at the right end.

Lab^* یعنی call string ها. سمت راست آن، آخرین فراخوانی است.

- $\hat{\iota}(\delta) = \begin{cases} \iota & \text{if } \delta = \Lambda \text{ (رشته خالی)} \\ \bot & \text{otherwise} \end{cases}$ ← شروع برنامه

Example -

مجموعه call string ها:

$\Lambda, [9], [9,4], [9,6], [9,4,4], [9,4,6], [9,6,6], \ldots$

آخرین فراخوانی

فراخوانی قبل از ۹

این اطلاعات می‌تواند به عنوان context نگه داشته شود.

(برنامه وضعیت ۱۱۸)

Subject
Date

برای procedure call

$(l_c, l_n, l_x, l_r) \in IF$

$[\text{call } p(a,z)]^{l_c}_{l_r}$

$\hat{f}^1_{l_c}(\hat{i})([\delta, l_c]) = f^1_{l_c}(\hat{i}(\delta))$ $\quad f^1_{l_c} : L \to L$

δ : append l_c

نمی‌دانیم چه وقت وارد بدنه رویه می‌شویم.

$$\hat{f}^1_{l_c}(\hat{i})(\delta') = \begin{cases} f^1_{l_c}(\hat{i}(\delta)) & \delta' = [\delta, l_c] \\ \bot & \text{otherwise} \end{cases}$$

otherwise برای حالتی که نمی‌دانیم

$f^1_{l_c}$: اطلاعاتی که می‌خواهیم وارد بدنه رویه شود به چه اطلاعاتی متناظر است.

$$\hat{f}^2_{l_c, l_r}(\hat{i}, \hat{i}')(\delta) = f^2_{l_c, l_r}(\hat{i}(\delta), \hat{i}'([\delta, l_c]))$$

$f^2_{l_c, l_r} : L \times L \to L$

اطلاعاتی که می‌خواهیم از بدنه رویه به نقطه فراخوانی برگردد

data flow اطلاعاتی که می‌خواهیم

Subject PA
Date

Context Sensitive: $f^2(\hat{l}(\delta), \hat{l}'([\delta, l_c]))$ ... l_c, l_r = =

(2.38 ...)

(context) ... eg.

... $\hat{L}$ ...

... وجود دارد.

* ... inter_procedural ...:

$$\hat{L} = \Delta \xrightarrow{\text{context}} L$$

for all labels except l_c, l_r $\quad A_\bullet(l) = \hat{f}_l(A_0(l))$

for all labels $\quad A_0(l) = \bigsqcup \{ A_\bullet(l') \mid (l', l) \in F \text{ or } (l'; l) \in F \} \sqcup \hat{i}^l_=$

$A_\bullet(l), A_0(l) \in \hat{L}$

$[call\ p(a,z)]^{l_c}_{l_r}$, $(l_c, l_n, l_x, l_r) \in IF$

... transfer ...

$\hat{f}_{l_n}(\hat{l}) = \hat{l}$, $\hat{f}_{l_x}(\hat{l}) = \hat{l}$

Subject
Date

$$\hat{f}^1_{l_c} : (\Delta \to L) \to (\Delta \to L)$$

$$A_\bullet(l_c) = \hat{f}^1_{l_c}(A_\bullet(l_c))$$

$$\hat{f}^2_{l_c, l_r} : (\Delta \to L) \times (\Delta \to L) \to (\Delta \to L)$$

$$A_\bullet(l_r) = \hat{f}^2_{l_c, l_r}(A_\bullet(l_c), A_\bullet(l_r))$$

specify کردن، transfer انتقال، signature تابع

اطلاعات ناقص را کامل می‌کند و داده‌ها:

$\Delta, \hat{f}^1_{l_c}, \hat{f}^2_{l_c, l_r}, \hat{l}_\varepsilon$ ← اینها به ترتیب نوع هستند.

(تابع کامل به همین شکل است)

Δ: Context هم اطلاعات مربوط به متغیرهای برنامه

call string ← تابع ای call pending

invalid مقدار خاص.

return تابع

PqPCO

l_c همان call string قبلی.

Subject PA
Date ۹۷/۳/۵

$$\hat{f}^{l}_{l_c}(\hat{l})([\delta, l_c]) = f^{l}_{l_c}(\hat{l}(\delta)) \quad , \quad f^{l}_{l_c} : L \to L$$

$$\hat{f}^{l}_{l_c}(\hat{l})(\delta') = \begin{cases} f^{l}_{l_c}(\hat{l}(\delta)) & \delta' = [\delta, l_c] \\ \bot & \text{otherwise} \end{cases}$$

$$\hat{f}^{2}_{l_c, l_r}(\hat{l}, \hat{l}') = f^{2}_{l_c, l_r}\left(\hat{l}(\delta), \hat{l}'([\delta, l_c])\right)$$

$$f^{2}_{l_c, l_r} : L \times L \to L$$

Subject
Date

$[call\ p(a,z)]^{l_c}_{l_r}$ ... Sign Detection (... Analysis

$\hat{L}_{Sign} = \Delta \to \wp(Var_* \to Sign)$ $\quad Sign = \{+,-,0\}$

isomorphic to $\wp(\Delta \times (Var_* \to Sign))$

$\hat{f}^{Sign1}_{l_c}(Z) = \bigcup_{union} \left\{ \{\delta'\} \times \phi^{Sign1}_{l_c}(\sigma^{Sign}) \,\middle|\, (\delta, \sigma^{Sign}) \in Z \wedge \delta' = [\delta, l_c] \right\}$

$\in \wp(\Delta \times (Var_* \to Sign))$

... Property, ... Context ...

... Context ...

$\phi^{Sign}_{l_c}(\sigma^{Sign}) = \left\{ \sigma^{Sign}[x \mapsto s][y \mapsto s'] \,\middle|\, s \in \Lambda_{Sign}[\![a]\!](\sigma^{Sign}) \wedge s' \in \{-,+,0\} \right\}$

... State ...

$\hat{f}^{Sign2}_{l_c l_r}(Z, Z') = \bigcup \left\{ \{\delta\} \times \phi^{Sign2}_{l_c l_r}(\sigma_1^{Sign}, \sigma_2^{Sign}) \,\middle|\, (\delta, \sigma_1^{Sign}) \in Z \wedge ([\delta, l_c], \sigma_2^{Sign}) \in Z' \right\}$

... Pending ...

Subject PA
Date

$$\phi^{Sig_2}_{lcalr}(\bar{\sigma}_1^{Sign}, \bar{\sigma}_2^{Sign}) = \{\bar{\sigma}_2^{Sign}[x \mapsto \bar{\sigma}_1^{Sign}(x), y \mapsto \bar{\sigma}_1^{Sign}(y), z \mapsto \bar{\sigma}_2^{Sign}(y)]\}$$

result

unbounded call string

if you consider call strings of bounded length,

$$\Delta = Lab^{\leq k}$$

$$\hat{c}(\delta) = \begin{cases} c & \text{if } \delta = \Lambda \\ \bot & \text{otherwise} \end{cases}$$

Call String $= \Lambda, [9], [4], [6]$, $k=1$

$\Delta = \Lambda, [9], [9,4], [9,6], [4,4], [4,6],$, $k=2$

$[6,4], [6,6]$

$[9,4,4]$, $k=\infty$

PqPCO

Subject

Date

bounded length ، روش کلی با محدودیت

$$\hat{f}^{l_c}(\hat{i})(\delta') = \bigsqcup \{ f^{l_c}(\hat{i}(\delta)) \mid \delta' = [\delta, l_c]_k \}$$

↓

call strings $[\delta, l_c]$

← ... δ ... but possibly truncated so as to have length at most k.

... $\delta \mapsto [\delta, l_c]_k$ ...

... injective ...

$$\hat{f}^{l_c, l_2}(\hat{i}, \hat{i}')(\delta) = f^{l_c, l_2}(\hat{i}(\delta), \hat{i}'([\delta, l_c]_k))$$

... bounded length ... Sign Detection ... 2.40 ...

... context ...

~~روش کلی برای تحلیل برنامه‌های ... call strings ... تحلیل ...~~

Subject PA
Date ۱۳۹۷، ۳، ۵

Shape Analysis حل می‌شود. ← ترجمه‌ی ... (قرن‌ها تعیین Π)

... call string ... Δ ... تحلیل‌های دیگر ... تحلیل

... control flow ... data flow ... تحلیل‌های ...

↓

... Assumption Set ... Labeled ...

... context ... abstract state ...

Assumption Sets :

... unbounded ، bounded ... small ، large ... (call string

large assumption set :

$L = \wp(D)$

$\hat{L} = \Delta \longrightarrow L$

isomorphic to $\wp(\Delta \times D)$

... context ... call ... (last call)

$\Delta = \wp(D)$

... call ...

(... context ...)

P4PCO

۱۴۵

Subject

Date

$\hat{\iota} = \{(\{\varepsilon\}, \iota)\}$

Detection of Sign Analysis

$\iota_{Sign} = [x \mapsto +, y \mapsto -, z \mapsto -]$

Possibilities:

$[x \mapsto +, y \mapsto 0, z \mapsto -]$, $[x \mapsto +, y \mapsto 0, z \mapsto 0]$, $[x \mapsto +, y \mapsto 0, z \mapsto +]$

$[x \mapsto +, y \mapsto +, z \mapsto -]$, $[x \mapsto +, y \mapsto +, z \mapsto 0]$, $[x \mapsto +, y \mapsto +, z \mapsto +]$

$\Delta = \wp(\text{abstract states mentioned above})$

$$\hat{f}^{l_c}_{l_c}(Z) = \bigcup \left\{ \{\delta'\} \times \phi^{l_r}_{l_c}(d) \;\middle|\; (\delta, d) \in Z \wedge \delta' = \{d'' \mid (\delta, d'') \in Z\} \right\}$$

$(l_c, l_n, l_y, l_r) \in \overline{IF}$, $[\text{call } p(a,z)]^{l_c}$

$\phi^{l_r}_{l_c} : D \to \wp(D)$

Subject PA
Date ۱۳۹۷/۲/۲۷

$$\hat{f}^{2}_{l_c, l_r}(z, z') = U\left\{ \{\delta\} \times \phi^{2}_{l_r, l_s}(d, d') \;\middle|\; (\delta, d) \in z \wedge (\delta', d') \in z' \wedge \cdots \right\}$$

$$\left\{ z \ni (p', \delta) \;\middle|\; p'' \right\} = \delta'$$

small assumption set ... large assumption sets ...

$$\Delta = D$$

monotone ... large assumption set ... small assumption set ... monotonicity ...

monotone ... small assumption set ...

$$\subseteq \left\{ d'' \;\middle|\; (s, d'') \in z \right\}$$

MFP ... MOP ... MFP

Subject PA
Date ۹۸/۴/۷

Flow Sensitivity and Insensitivity :

data flow و information flow ، ... ↓

$S_1 ; S_2$ ... Flow Sensitive ...

$S_2 ; S_1$ ...

$S_1 ; S_2$ ، $S_2 ; S_1$ ...

control flow ...

... Soundness ...

... Sound ...

... flow sensitive ... information flow ... binding ...
Security

... dynamic binding ...

Subject PA

Date ۹۷/۱۲/۱۷

upgrade / downgrade

flow insensitive

information flow, data flow

(ضعیف / قوی)

Sets of Assignments / Variables

P_* begin D_* S_* end

proc p (val x, res y) is S end in D_*

l_n l_x

global variable

IAV(p) : the set of global variables that might be assigned directly or indirectly when p is called.

$AV([skip]^l) = \emptyset$ $AV([x:=a]^l) = \{x\}$ ← directly assigned vars

$AV(S_1; S_2) = AV(S_1) \cup AV(S_2)$

$AV(\text{if } [b]^l \text{ then } S_1 \text{ else } S_2) = AV(S_1) \cup AV(S_2)$

Subject
Date

$$AV([\text{while } [b]^{l} \text{ do } S]) = AV(S)$$

$$AV\left(\underset{l_r}{[\text{call } p(a,z)]^{l_c}}\right) = \{z\}$$

called procedures

$$cp([\text{skip}]^{l}) = cp([x := a]^{l}) = \emptyset$$

$$cp(S_1; S_2) = cp(S_1) \cup cp(S_2)$$

$$cp(\text{if } [b]^{l} \text{ then } S_1 \text{ else } S_2) = cp(S_1) \cup cp(S_2)$$

$$cp(\text{while } [b]^{l} \text{ do } S) = cp(S)$$

$$cp\left(\underset{l_r}{[\text{call } p(a,z)]^{l_c}}\right) = \{p\}$$

[illegible] AV, cp [illegible] flow insensitive [illegible] (درجه ۱ آبان – امیرکبیر)

[illegible] S_* [illegible] ($main_*$). [illegible] $p' \in cp(S)$ [illegible] p [illegible] call graph [illegible].

PqPCO

Subject PA
Date ۱۳۹۷/۳/۷

$$IAV(P) = \left(AV(S) \setminus \{x\}\right) \cup \bigcup \left\{ IAV(P') \mid P' \in CP(S) \right\}$$

where proc $P(\overset{val}{x}, \overset{res}{y})$ is S end is in P_*

این تحلیل، تحلیلی flow insensitive خواهد بود و دلیل آن را در ادامه خواهیم دید.

در ضمن این تحلیل برای زبان‌های شیءگرا مناسب است.

Shape Analysis

هدف این تحلیل آن است که اطلاعاتی درباره ساختار heap به دست آوریم. مثلاً اینکه آیا null pointer dereference رخ می‌دهد یا خیر.

همچنین این تحلیل برای بررسی aliasing، race conditions و ... کاربرد دارد.

انواع تحلیل‌ها در Shape Analysis و Pointer Analysis متفاوت است و در ادامه بررسی خواهیم کرد.

* Static Single Assignment (SSA): در این نوع تحلیل، هر متغیر در برنامه تنها یک بار مقداردهی می‌شود. (با استفاده از rename)

(ادامه در جلسه بعد)

Subject PA
Date ۹۷/۲/۱۲

(تصویر)

Constraint-Based Analysis : (کنترل مبتنی بر محدودیت)

این روش برای کنترل جریان داده‌ها در برنامه‌ها به کار می‌رود. flow، flow^R، inter-flow و given را محاسبه می‌کند.

این روش برای زبان‌های imperative مانند C و Pascal و همچنین object-oriented و functional که از procedure استفاده می‌کنند مناسب است.

در این روش ابتدا control flow برنامه را پیدا می‌کنیم که معمولاً trivial است، سپس با استفاده از control flow داده‌ها را پیدا می‌کنیم.

این روش برای زبان‌های functional مناسب نیست زیرا در آن‌ها function به عنوان پارامتر به functional دیگر داده می‌شود.

Computational Model = Values + Operations
↓ ↓ ↓
Functional ⟹ Functions + Function Applications

pure Functional ← Haskell
(based on λ-calculus)

```
let f = fn x => x 1;
    g = fn y => y + 2;
    h = fn z => z + 3 g
in (f g) + (f h)
```

Subject PA
Date ۹۷/۴/۱۲

dynamic dispatch → procedure

FUN langrage :

Syntactic Category

$e \in Exp \longrightarrow$

$t \in Term \longrightarrow$

$f, x \in Var$

$c \in Const$

$op \in Operations$

$l \in Lab$

$e ::= t^{l}$

$t ::= c \mid x \mid fn\ x \Rightarrow e_0 \mid$

$fun\ f\ x \Rightarrow e_0 \mid e_1\ e_2 \mid$

$if\ e_0\ then\ e_1\ else\ e_2 \mid$

$let\ x = e_1\ in\ e_2 \mid e_1\ op\ e_2$

۱۵۳

Subject

Date

free variables

$FV : (Term \cup Exp) \rightarrow Var$

$FV(fn\ x \Rightarrow e_0) = FV(e_0) \setminus \{x\}$

$\therefore$ bound (or bind), $\therefore$ free i.e., x

$FV(fun\ f\ x \Rightarrow e_0) = FV(e_0) \setminus \{x, f\}$

$FV(let\ x = e_1\ in\ e_2) = FV(e_1) \cup (FV(e_2) \setminus \{x\})$

[$\because$ visible free (e_1 never sees x)]

Example -

I. $\left((fn\ x \Rightarrow x^1)^2\ (fn\ y \Rightarrow y^3)^4\right)^5$

II. $(let\ g = (fun\ f\ x \Rightarrow (f^1\ (fn\ y \Rightarrow y^2)^3)^4)^5\ in$

$(g^6\ (fn\ z \Rightarrow z^7)^8)^9)^{10}$

$\therefore$ infinite loop (II run)

Subject PA
Date ۱۳۹۷/۳/۱۲

Control Flow Analysis :

I. 0-CFA :

(Zero)

Abstract Cache

Abstract Environment

$\hat{v} \in \hat{Val}$ $\quad \hat{Val} = \hat{\rho}(Term)$

(abstract values)

$fn\ x \Rightarrow e_0$, $fun\ f\ x \Rightarrow e_0$

$\hat{e} \in \hat{Env} = Var \rightarrow \hat{Val}$

$\hat{c} \in \hat{Cache} = Lab \rightarrow \hat{Val}$

Subject PA
Date

(کنترل نشان می‌دهد به چه توابعی می‌خواهد اعمال شود.)

Example - $\left((fn\ x \Rightarrow x^4)^2 (fn\ y \Rightarrow y^3)^4 \right)^5$

	کلیل اول $(\hat{C}_e, \hat{\rho}_e)$	کلیل دوم $(\hat{C}'_e, \hat{\rho}'_e)$	کلیل سوم $(\hat{C}''_e, \hat{\rho}''_e)$
1	$\{fn\ y \Rightarrow y^3\}$	$\{fn\ y \Rightarrow y^3\}$	$\{fn\ x \Rightarrow x^4\}, \{fn\ y \Rightarrow y^3\}$
2	$\{fn\ x \Rightarrow x^4\}$	$\{fn\ x \Rightarrow x^4\}$	〃
3	ϕ	ϕ	〃
4	$\{fn\ y \Rightarrow y^3\}$	$\{fn\ y \Rightarrow y^3\}$	〃
5	$\{fn\ y \Rightarrow y^3\}$	$\{fn\ y \Rightarrow y^3\}$	〃
x	$\{fn\ y \Rightarrow y^3\}$	ϕ	〃
y	ϕ	ϕ	〃

↑ کلیل سوم

کلیل اول و سوم acceptable هستند اما کلیل سوم دقیق‌تر است.

بنابراین کلیل acceptable هم دقیق است!
(precise)

Use-Definition chain, Control Flow Analysis

PAPCO

Subject
Date ۹۷/۲/۱۲

Subject

Date

4

the most challenging rule

$(\hat{c},\hat{\rho}) \models (t_1^{l_1}\ t_2^{l_2})^{l}$ iff $(\hat{c},\hat{\rho}) \models t_1^{l_1} \wedge (\hat{c},\hat{\rho}) \models t_2^{l_2} \wedge$

$(\forall (fn\ x \Rightarrow t_0^{l_0}) \in \hat{c}(l_1).$

$(\hat{c},\hat{\rho}) \models t_0^{l_0} \wedge \hat{c}(l_2) \subseteq \hat{\rho}(x) \wedge$

$\hat{c}(l_0) \subseteq \hat{c}(l)) \wedge$

$(\forall (fun\ f\ x \Rightarrow t_0^{l_0}) \in \hat{c}(l_1).\ (\hat{c},\hat{\rho}) \models t_0^{l_0} \wedge \hat{c}(l_2) \subseteq \hat{\rho}(x) \wedge \hat{c}(l_0) \subseteq \hat{c}(l) \wedge \{fun\ f\ x \Rightarrow t_0^{l_0}\} \subseteq \hat{\rho}(f))$

$(\hat{c},\hat{\rho}) \models (\text{if } t_0^{l_0} \text{ then } t_1^{l_1} \text{ else } t_2^{l_2})^{l}$

iff $(\hat{c},\hat{\rho}) \models t_0^{l_0} \wedge (\hat{c},\hat{\rho}) \models t_1^{l_1} \wedge (\hat{c},\hat{\rho}) \models t_2^{l_2}$

$\wedge\ \hat{c}(l_1) \subseteq \hat{c}(l) \wedge \hat{c}(l_2) \subseteq \hat{c}(l)$

$(\hat{c},\hat{\rho}) \models (\text{let } x = t_1^{l_1} \text{ in } t_2^{l_2})^{l}$

iff $(\hat{c},\hat{\rho}) \models t_1^{l_1} \wedge (\hat{c},\hat{\rho}) \models t_2^{l_2} \wedge$

$\hat{c}(l_1) \subseteq \hat{\rho}(x) \wedge \hat{c}(l_2) \subseteq \hat{c}(l)$

$(\hat{c},\hat{\rho}) \models (t_1^{l_1}\ \text{op}\ t_2^{l_2})$ iff $(\hat{c},\hat{\rho}) \models t_1^{l_1} \wedge (\hat{c},\hat{\rho}) \models t_2^{l_2}$

Subject PA
Date ۹۷/۴/۱۹

مثلاً کلاس برای $e = \left((fn\ x \Rightarrow x')^2 (fn\ y \Rightarrow y^3)^4\right)^5$ صفحه ۱۵۶ کتاب کلمن اول

سوال این است که آیا $(\hat{c}_e, \hat{\rho}_e) \vdash e$ هست؟ باید از این قواعد acceptability سوال کنیم

$$\hat{c}_e(2) = \{fn\ x \Rightarrow x'\}$$

باید این موارد برقرار باشد:

$$(\hat{c}_e, \hat{\rho}_e) \vdash (fn\ x \Rightarrow x')^2 \wedge$$

$$(\hat{c}_e, \hat{\rho}_e) \vdash (fn\ y \Rightarrow y^3)^4 \wedge$$

$$(\hat{c}_e, \hat{\rho}_e) \vdash x' \wedge \hat{c}_e(4) \subseteq \hat{\rho}_e(x) \wedge \hat{c}_e(1) \subseteq \hat{c}_e(5)$$

به همین کلاس کتاب اول برمیگردیم

با این قواعدی که نوشتیم هم میتوانیم چک کنیم یک کلاس خوب هست یا نه و هم اینکه خودش میتواند یک کلاس را بسازد

چون recursive محدود نمی‌کند، راحتی زیاد، دستگاهی از constraint حل میکند.

میتوان $(\hat{c}, \hat{\rho}) \vdash e$ را تعریف کرد:

$$\vdash : (\widehat{Cache} \times \widehat{Env} \times Exp) \longrightarrow \{true, false\}$$

قاعده‌ای برای acceptability نوشته شد

P4PCO

۵۹

Subject
Date

left_hand side
right_hand side

lhs ⊆ rhs

lhs ⊆ rhs

$\hat{c}(l), \hat{\rho}(x), \{t\}$ $\hat{c}(l), \hat{\rho}(x)$

term

Subject
Date ۹۷/۳/۱۹

well-definedness ←

Well-definedness of the Analysis:

Subject
Date

نحوه تعریف قواعد در تعاریف استقرایی هم همین‌طور است اما تعبیر و بررسی صفات co-inductive

اندازه استقرایی نداشتیم!

هر مجموعه functional متناظر با یک رابطه

functional ای که least fixed point ، متناظر inductive ، و greatest fixed point ، coinductive

$$\models \quad Q : \Big((\hat{Cache} \times \hat{Env} \times Exp) \to \{true, false\} \Big)$$

functional

$$\to \Big((\hat{Cache} \times \hat{Env} \times Exp) \to \{true, false\} \Big)$$

let : حالت دیگر قاعده

$$Q(R)\left(\hat{C}, \hat{\rho}, (\text{let } x = t_1^{l_1} \text{ in } t_2^{l_2})^{l}\right) =$$

$$R(\hat{C}, \hat{\rho}, t_1^{l_1}) \wedge R(\hat{C}, \hat{\rho}, t_2^{l_2}) \wedge \hat{C}(l_1) \subseteq \hat{\rho}(x) \wedge \hat{C}(l_2) \subseteq \hat{C}(l)$$

بنابراین:

$$Q(R) = R'$$

$Q(R) = R$ به دنبال نقطه fixed point

PqPCO

Subject PA
Date ۹۷/۲/۱۹

$R_1 \sqsubseteq R_2$ iff $\forall (\hat{c}, \hat{\rho}, e)$.

$R_1(\hat{c}, \hat{\rho}, e) = true \rightarrow R_2(\hat{c}, \hat{\rho}, e) = true$

complete lattice

bot

false

gfp, lfp

Tarsky

Q is monotone

F = the greatest fixed point of Q

well_definedness

gfp = lfp

co-induction

co-induction

derivation

Subject

Date 97/3/21

(هنوز 0-CFA را برای مثال بررسی می‌کنیم.)

* روش سوال بعدی است: تعدادی از تحلیل‌ها well-defined هستند.

1) چه تحلیل‌هایی Semantically correct، Safe approximation هستند؟

2) از بین تحلیل‌های ممکن، بهترین را چطور پیدا کنیم؟ (Subject Reduction Result)

existence of least solutions

تحلیل‌هایی که Preserve می‌کنند

Semantic reductions

یعنی reduction بعد از reduction چه اتفاقی می‌افتد؟

[Theory ... Type ... Subject Reduction Result]

progress ? preservation ? evaluation بعد از term

Type Safety

Semantics ، Semantic Correctness ، FUN

Substitution ، λ-calculus ، FUN

abstract value

closure) FUN

closure ... Value

Subject PA
Date ۹۷,۲,۱۱

close t in ρ → local env

closure of function t

(env) $\rho ::= [\,] \mid \rho[x \mapsto v]$

to value $v ::= c \mid \text{close } t \text{ in } \rho$

closure: intermediate term, expression

$\rho \vdash ie \to ie$ (intermediate exp)

$$\rho \vdash (\text{fn } x \Rightarrow e_0)^{l} \to (\text{close } (\text{fn } x \Rightarrow e_0) \text{ in } \rho_0)^{l}$$

where $\rho_0 = \rho \mid FV(\text{fn } x \Rightarrow e_0)$

FV

$$\frac{\rho \vdash ie_1 \to ie_1'}{\rho \vdash (ie_1 ; ie_2)^{l} \to (ie_1' ; ie_2)^{l}} \qquad \frac{\rho \vdash ie_2 \to ie_2'}{\rho \vdash (v_1^{l_1} ; ie_2)^{l} \to (v_1^{l_1} ; ie_2')^{l}}$$

$$\rho \vdash \left((\text{close } (\text{fn } x \Rightarrow e_1) \text{ in } \rho_1)^{l_1} \; v_2^{l_2} \right)^{l} \to (\text{bind } \rho_1[x \mapsto v_2] \text{ in } e_1)^{l}$$

(Substitution)

Subject PA
Date ۱۳۹۷/۳/۲۱

(جلسه سوم - برگزاری)

برای بررسی acceptability یک intermediate expression

$(\hat{c}, \hat{\rho}) \models (\text{bind } \rho \text{ in } it_0^{l_0})^{l}$ iff $(\hat{c}, \hat{\rho}) \models it_0^{l_0} \wedge$

$\hat{c}(l_0) \subseteq \hat{c}(l) \wedge$

$\rho \; R \; \hat{\rho}$

رابطه بین دو محیط

$(\hat{c}, \hat{\rho}) \models (\text{close } t_0 \text{ in } \rho)^{l}$ iff $\{t_0\} \subseteq \hat{c}(l) \wedge \rho R \hat{\rho}$

رابطه بین دو محیط $\rho \; R \; \hat{\rho}$ iff $dom(\rho) \subseteq dom(\hat{\rho}) \wedge$

$\forall x \in dom(\hat{\rho}) . \forall t_x . \forall \rho_x .$

$(\rho(x) = \text{close } t_x \text{ in } \rho_x) \Rightarrow (t_x \in \hat{\rho}(x) \wedge \rho_x R \hat{\rho})$

قضیه: اگر $\rho R \hat{\rho}$ و $\rho \vdash ie \to ie'$ آنگاه $(\hat{c}, \hat{\rho}) \models ie$ نتیجه می‌دهد $(\hat{c}, \hat{\rho}) \models ie'$.

Theorem - If $\rho R \hat{\rho}$ and $\rho \vdash ie \to ie'$,

then $(\hat{c}, \hat{\rho}) \models ie$ implies $(\hat{c}, \hat{\rho}) \models ie'$.

~~(اثبات صحت subject reduction برای این تعریف)~~

[اثبات صحت subject reduction برای این تعریف در جلسه بعد ارائه می‌شود.]

Subject PA
Date ۹۷/۴/۲۱

ک اب ریاضی : سوال دوم (در مورد least solutions) ۶

$(\hat{c}_1, \hat{\rho}_1) \sqsubseteq (\hat{c}_2, \hat{\rho}_2)$ iff $\forall l \in Lab.\ \hat{c}_1(l) \subseteq \hat{c}_2(l) \wedge$

$\forall x \in Var.\ \hat{\rho}_1(x) \subseteq \hat{\rho}_2(x)$

بین ۲ دو کیل acceptable (یعنی Safe approx.) ، این ابطه یک کیل کمتر را به کیل بزرگتر ربط میدهد.

- A subset Y of a complete lattice $L = (L, \sqsubseteq)$ is a Moore family if and only if $\bigsqcap Y' \in Y$ for all $Y' \subseteq Y$.

(meet هر زیرمجموعه از آن در خود مجموعه است.)

meet هر زیرمجموعه از Moore family ، acceptable ها هم کیل هایی هستند.

کیل ها هم least هستند.
(کمترین شان)

Proposition — For all $ie \in IExp$, the set $\{(\hat{c}, \hat{\rho}) \mid (\hat{c}, \hat{\rho}) \models ie\}$ is a Moore Family.

بنابراین meet تمام کیل هایی که میتوان ساخت هم least solution است.

Subject
Date

(آزمون فصل سوم، کمپایلر و تحلیل معنایی)

اگر meet با solution ها برابر باشد، برنامه ما درست کار می‌کند!

الگوریتم‌های تحلیل معنایی به دو دسته تقسیم می‌شوند:

1) Syntax-directed analysis
2) Constraint-Based analysis

→ معیار ما acceptability است!

تعریف و استدلال نوشته.

(آزمون!)