Topici speciale în logică și securitate I

Analiza fluxului de date

Paul Irofti și Ioana Leuștean

Master anul II, Sem. I, 2019-2020

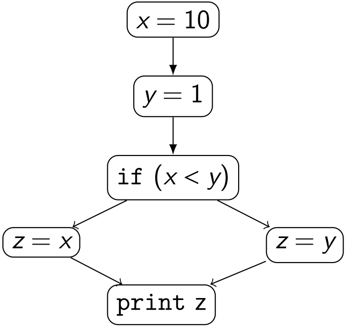
Ce este analiza statică?

Analiza statică presupune analiza proprietăților programelor fără a le rula.

Exemple de proprietăți: analiza tipurilor, pointeri nuli, atribuiri nefolosite, vulnerabilităti de cod (de exemplu depășiri de indici), etc.

Control Flow Graphs (CFG)

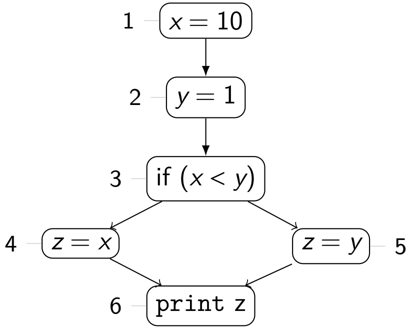
```
x = 10;
y = 1;
if  (x < y) z = x;
     else   z = y;
print z
```



Control Flow Graphs (CFG)

```
x = 10;
y = 1;
if  (x < y) z = x;
     else   z=y;
print z
```

Adăugăm etichete pentru nodurile grafului.



Exemplu: determinarea variabilelor *live*

https://cs420.epfl.ch/c/06_dataflow-analysis.html

O variabilă este activă (live) într-un punct al programului dacă este posibil ca valoarea curentă variabilei să fie citită înainte de a fi rescrisă.

Pentru un nod n din CFG vom nota cu v_n mulțimea variabilelor care sunt *live* înaintea lui.

Observăm că

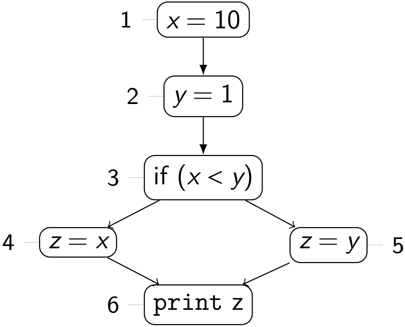
$$v_n = \Big(\bigcup \{v_i \mid i \text{ succesor al lui } n\} \setminus Written(n)\Big) \cup Read(n)$$

$Written(n)$ este mulțimea variabilelor care primesc valori în nodul n,
$Read(n)$ este mulțimea variabilelor care sunt citite în nodul n.

Exemplu: determinarea variabilelor *live*

Ținând cont de condiția găsită obținem un sistem de constrângeri:

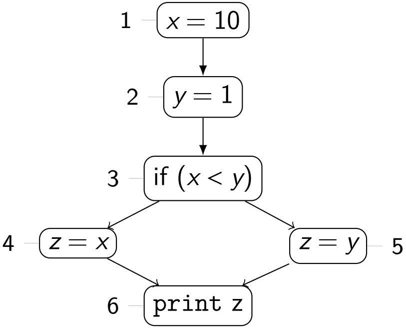
$$
\begin{array}{lcl}
v_1 & = & v_2 \setminus \{x\} \\
v_2 & = & v_3 \setminus \{y\} \\
v_3 & = & v_4 \cup v_5 \cup \{x, y\} \\
v_4 & = & \left(v_6 \setminus \{z\}\right) \cup \{x\} \\
v_5 & = & \left(v_6 \setminus \{z\}\right) \cup \{y\} \\
v_6 & = & \{z\}
\end{array}
$$



Exemplu: determinarea variabilelor *live*

Rezolvând sistemul, găsim o soluție:

$$\begin{aligned} v_1 &= \emptyset \\ v_2 &= \{x\} \\ v_3 &= \{x, y\} \\ v_4 &= \{x\} \\ v_5 &= \{y\} \\ v_6 &= \{z\} \end{aligned}$$



CFG - observații

- În analiza problemei anterioare, fiecărui nod din graful de control al fluxului i-am asociat o mulțime de variabile, ce reprezintă soluția unui sistem de ecuații in $\mathcal{P}(Var)$, unde Var este mulțimea variabilelor din program.

- În general, analiza unei probleme poate conduce la un sistem de ecuații într-o latice oarecare L. Există o metodă generală de rezolvare a unui astfel de sistem?

Laticea produs

Fie L o latice și $n \geq 1$.

Laticea produs L^n

Pe L^n definim relația de ordine pe componente:

$$(x_1, \ldots, x_n) \leq (y_1, \ldots, y_n) \text{ ddacă } x_i \leq y_i \text{ pt. orice } i \in \{1, \ldots, n\}$$

Observăm că $(L^n, \leq)$ este latice. În plus, dacă L este latice completă atunci și L^n este latice completă.

Sisteme de ecuații în latici complete

Fie L o latice completă, $n \geq 1$ și
$F_1, \ldots, F_n : L^n \to L$ funcții monotone.
Vrem să rezolvăm sistemul

$$\begin{array}{lcl} x_1 & = & F_1(x_1, \ldots, x_n) \\ x_2 & = & F_2(x_1, \ldots, x_n) \\ & \ldots & \\ x_n & = & F_n(x_1, \ldots, x_n) \end{array}$$

cu $x_1, \ldots, x_n \in L$.

Definim funcția $F : L^n \to L^n$ prin

$$F(x_1, \ldots, x_n) = (F_1(x_1, \ldots, x_n), \ldots, F_1(x_1, \ldots, x_n))$$

și observăm că sistemul se poate scrie

$$F(x_1, \ldots, x_n) = (x_1, \ldots, x_n)$$

Sistemul de ecuații poate fi rezolvat folosind Teorema de punct fix!

Determinarea variabilelor *live*

În consecință, a rezolva sistemul:

$$
\begin{aligned}
v_1 &= v_2 \setminus \{x\} \\
v_2 &= v_3 \setminus \{y\} \\
v_3 &= v_4 \cup v_5 \cup \{x, y\} \\
v_4 &= \big(v_6 \setminus \{z\}\big) \cup \{x\} \\
v_5 &= \big(v_6 \setminus \{z\}\big) \cup \{y\} \\
v_6 &= \{z\}
\end{aligned}
$$

revine la a găsi o soluție a ecuației:

$$F(x_1, x_2, x_3, x_4, x_5, x_6) =$$
$$\big(x_2 \setminus \{x\}\,,\ x_3 \setminus \{y\}\,,\ x_4 \cup x_5 \cup \{x, y\}\,,\ \big(x_6 \setminus \{z\}\big) \cup \{x\}\,,\ \big(x_6 \setminus \{z\}\big) \cup \{y\}\,,\ \{z\}\big)$$

în laticea completă $(\mathcal{P}(Var), \subseteq, \emptyset, Var)$.

Rezolvarea folosind teorema de punct fix

$F(x_1, x_2, x_3, x_4, x_5, x_6) =$
$(x_2 \setminus \{x\}\,,\ x_3 \setminus \{y\}\,,\ x_3 \setminus \{y\}\,,\ (x_6 \setminus \{z\}) \cup \{x\}\,,\ (x_6 \setminus \{z\}) \cup \{y\}\,,\ \{z\})$

Determinarea celui mai mic punct fix:

	x_1	x_2	x_3	x_4	x_5	x_6
$\perp$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$F(\perp)$	$\emptyset$	$\emptyset$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$
$F^2(\perp)$	$\emptyset$	$\{x\}$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$
$F^3(\perp)$	$\emptyset$	$\{x\}$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$

unde $\perp = (\emptyset, \ldots, \emptyset)$

IMP și variabile *live*

Fie setul variabilelor succesoare nodului n

$$Join(v_n) = \bigcup \{v_i \mid i \text{ succesor al lui } n\}$$

Formula generală pentru variabilele *live* în nodul k este:

$$v_n = Join(v_n) \setminus Written(n) \cup Read(n)$$

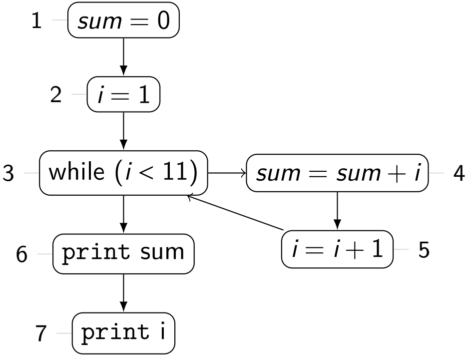
Formele particulare instrucțiunilor IMP:

- Expresii (E) — E — `x + 3, (x > 7)`
- Blocuri de instrucțiuni
  - De atribuire — `x = E;` — $v_n = Join(v_n) \setminus \{x\} \cup Var(E)$
  - Condiționale — `if (E)` — $v_n = Join(v_n) \cup Var(E)$
  - De ciclare — `while (E)` — $v_n = Join(v_n) \cup Var(E)$
  - De tipărire — `print E;` — $v_n = Join(v_n) \cup Var(E)$

unde $Var(E)$ reprezintă setul variabilelor prezente în expresia E.

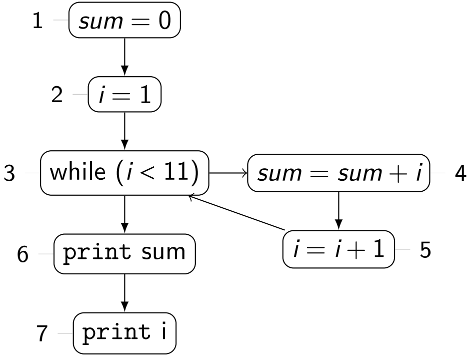
Control Flow Graphs (CFG)

```
sum = 0;
i = 1;
while (i < 11) {
    sum = sum + i;
    i = i + 1;
}
print sum
print i
```



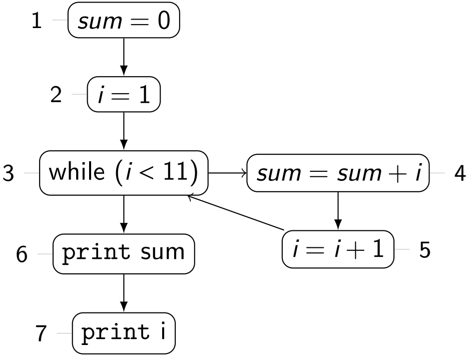
Control Flow Graphs (CFG)

$$
\begin{aligned}
v_1 &= v_2 \setminus \{sum\} \\
v_2 &= v_3 \setminus \{i\} \\
v_3 &= v_4 \cup v_6 \cup \{i\} \\
v_4 &= \left(v_5 \setminus \{sum\}\right) \cup \{i, sum\} \\
v_5 &= \left(v_3 \setminus \{i\}\right) \cup \{i\} \\
v_6 &= v_7 \cup \{sum\} \\
v_7 &= \{i\}
\end{aligned}
$$



Control Flow Graphs (CFG)

$$
\begin{aligned}
v_1 &= \emptyset \\
v_2 &= \{sum\} \\
v_3 &= \{i, sum\} \\
v_4 &= \{i, sum\} \\
v_5 &= \{i, sum\} \\
v_6 &= \{i, sum\} \\
v_7 &= \{i\}
\end{aligned}
$$



Rezolvarea folosind teorema de punct fix

$F(x_1, x_2, x_3, x_4, x_5, x_6, x_7) =$
$(x_2 \setminus \{sum\}, x_3 \setminus \{i\}, x_4 \cup x_6 \cup \{i\}, (x_5 \setminus \{sum\}) \cup \{i, sum\}, x_3 \setminus \{i\}) \cup \{i\}, x_7 \cup \{sum\}, \{i\})$

Determinarea celui mai mic punct fix:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
$\perp$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$F(\perp)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{sum\}$	$\{i\}$
$F^2(\perp)$	$\emptyset$	$\emptyset$	$\{sum, i\}$	$\{sum, i\}$	$\emptyset$	$\{sum\}$	$\{i\}$
$F^3(\perp)$	$\emptyset$	$\{sum\}$	$\{sum, i\}$	$\{sum, i\}$	$\{sum, i\}$	$\{sum\}$	$\{i\}$

unde $\perp = (\emptyset, \ldots, \emptyset)$

Exercițiu

```
x = 42;
while (x > 1) {
    y = x / 2;
    if (y > 3)
        x = x - y;
    z = x - 4;
    if (z > 0)
        x = x / 2;
    z = z - 1;
}
print x
```

Algoritmi pentru determinarea *lfp*

Pentru a obține cel mai mic punct fix (*lfp*) folosim lanțul până obținem convergența

$$\mathbf{F}^0(\bot) = \bot \leq \mathbf{F}(\bot) \leq \mathbf{F}^2(\bot) \leq \cdots \leq \mathbf{F}^n(\bot) \leq \cdots$$

Ceea ce poate fi pus în pseudo-cod în forma:

Algorithm 1: NaiveLFP

```
x = (⊥, ⊥, ..., ⊥);
while x ≠ F(x) do
    x = F(x);
return x;
```

Complexitate: depinde de înălțimea laticei L^n și evaluarea lui $\boldsymbol{F}(\boldsymbol{x})$.

Convergență: sunt necesare n iterații.

Algoritmi pentru determinarea *lfp*

Algoritmul 1 nu profită de structura laticei:

- recalculează intrări neschimbate de la ultima iterație
- nu ia în considerare intrările deja calculate la iterația curentă

Pentru primul exemplu

$F(x_1, x_2, x_3, x_4, x_5, x_6) =$
$(x_2 \setminus \{x\}\,,\ x_3 \setminus \{y\}\,,\ x_4 \cup x_5 \cup \{x, y\}\,,\ (x_6 \setminus \{z\}) \cup \{x\}\,,\ (x_6 \setminus \{z\}) \cup \{y\}\,,\ \{z\})$

punctul fix ar fi calculat în 6 iterații

	x_1	x_2	x_3	x_4	x_5	x_6
$\perp$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$F(\perp)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{z\}$
$F^2(\perp)$	$\emptyset$	$\emptyset$	$\emptyset$	$\{x\}$	$\{y\}$	$\{z\}$
$F^3(\perp)$	$\emptyset$	$\emptyset$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$
$F^4(\perp)$	$\emptyset$	$\{x\}$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$
$F^5(\perp)$	$\emptyset$	$\{x\}$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$

Comparație algoritmi *lfp*

Ideal un algoritm eficient ar reduce numărul de iterații:

	x_1	x_2	x_3	x_4	x_5	x_6
$\perp$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$F(\perp)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{z\}$
$F^2(\perp)$	$\emptyset$	$\emptyset$	$\emptyset$	$\{x\}$	$\{y\}$	$\{z\}$
$F^3(\perp)$	$\emptyset$	$\emptyset$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$
$F^4(\perp)$	$\emptyset$	$\{x\}$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$
$F^5(\perp)$	$\emptyset$	$\{x\}$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$

versus

	x_1	x_2	x_3	x_4	x_5	x_6
$\perp$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$F(\perp)$	$\emptyset$	$\emptyset$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$
$F^2(\perp)$	$\emptyset$	$\{x\}$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$
$F^3(\perp)$	$\emptyset$	$\{x\}$	$\{x, y\}$	$\{x\}$	$\{y\}$	$\{z\}$

Round Robin *lfp*

Actualizare pe componente tip Gauss-Seidel

- calculează x_i în funcție de $x_{j<i}$ de la iterația curentă
- *lfp* este atins deși o iterație mare produce rezultate diferite față de Alg. 1

Algorithm 2: RoundRobinLFP

$(x_1, x_2, \ldots, x_n) = (\perp, \perp, \ldots, \perp)$;
while $(x_1, x_2, \ldots, x_n) \neq \boldsymbol{F}(x_1, x_2, \ldots, x_n)$ **do**
  for $i \in 1 \to n$ **do**
    $x_i = \boldsymbol{F}_i(x_1, x_2, \ldots, x_n)$;
return $(x_1, x_2, \ldots, x_n)$;

Complexitate: depinde de înălțimea laticei L^n și evaluarea lui $\boldsymbol{F}_i(\boldsymbol{x})$.

Convergență: sunt necesare cel mult n iterații.

Chaotic *lfp*

Actualizare pe componente în ordine aleatoare

- ordinea operațiilor nu contează în Alg. 2
- calculează x_i în funcție de x_j calculați la iterația curentă; fără o ordine anume
- Alg. 2 recalculează în continuare intrările neschimbate de la ultima iterație
- constrângerile trebuie aplicate până la convergență

Algorithm 3: ChaoticLFP

$(x_1, x_2, \ldots, x_n) = (\bot, \bot, \ldots, \bot)$;
while $(x_1, x_2, \ldots, x_n) \neq \boldsymbol{F}(x_1, x_2, \ldots, x_n)$ **do**
  $i = random(1 \to n)$ $x_i = \boldsymbol{F}_i(x_1, x_2, \ldots, x_n)$;
return $(x_1, x_2, \ldots, x_n)$;

Complexitate: depinde de în primul rând de cum este ales i la fiecare iterație, apoi de înălțimea laticei L^n și evaluarea lui $\boldsymbol{F}_i(\boldsymbol{x})$.

Convergență: algoritmul nu este garantat că se oprește, dar dacă se oprește rezultatul este corect

Fie $dep(v)$ setul de noduri a căror informație depinde de informația din nodul v.

Algorithm 4: SimpleWorkListLFP

$(x_1, x_2, \ldots, x_n) = (\perp, \perp, \ldots, \perp)$;
$W = \{v_1, v_2, \ldots, v_n\}$;
while $W \neq \emptyset$ **do**
    $v_i = W.removeNext()$;
    $y = \boldsymbol{F}_i(x_1, x_2, \ldots, x_n)$;
    if $y \neq x_i$ **then**
        $x_i = y$;
        for $v_j \in dep(v_i)$ **do**
            W.add(v_j);
return $(x_1, x_2, \ldots, x_n)$;

Funcțiile *removeNext*() și *add*() aleg un nod aleator din W și, respectiv, adaugă un nod în W.

SimpleWorkList *lfp*

Proprietăți

- calculează doar constrângerile necesare nodului curent
- nu recaclulează intrărirle neschimbate de la ultima iterație
- calculează x_i în funcție de x_j calculați la iterația curentă; fără o ordine anume
- constrângerile sunt aplicate până la convergență
- fiecare iterație are același efect ca o iterație a Alg. 3

Complexitate: depinde de numărul de noduri n, de înălțimea laticei h, și evaluarea lui $\boldsymbol{F}_i(\boldsymbol{x})$.

Convergență: o iterație fie scade volumul de muncă din W fie urcă pe latice; algoritmul se încheie pentru că înălțimea laticei este finită iar iterațiile se opresc când W este vid.