Překladače (5. přednáška)

Jan Hubička

Katedra aplikované matematiky
Univerzita Karlova
Praha

23. března 2020

Minulá přednáška

Dataflow je technika jak řešit různé problémy v rámci celé funkce (implementovat globální optimalizace).
Většinou se implementuje jako relaxační algoritmus, který pracuje nad dobře definovaným polosvazem hodnot konečné hloubky od optimistického řešení k fixnímu bodu.

Minulá přednáška

Dataflow je technika jak řešit různé problémy v rámci celé funkce (implementovat globální optimalizace).
Většinou se implementuje jako relaxační algoritmus, který pracuje nad dobře definovaným polosvazem hodnot konečné hloubky od optimistického řešení k fixnímu bodu.

1. Globální propagace konstant
2. Mazání mrtvého kódu (za pomoci liveness dataflow)
3. Globální eliminace společných podvýrazů
4. ...

Propagace konstant nad celou funkcí

Definition (Plná redundance)

Výskyt výrazu E na pozici P je redundantní pokud je dostupný (available) před jeho vyhodnocením.

Výraz je dosuptný pokud na každé cestě z Entry do P je E vypočtený a hodnoty nejsou následně změneny.

Propagace konstant nad celou funkcí

Definition (Plná redundance)

Výskyt výrazu E na pozici P je redundantní pokud je dostupný (available) před jeho vyhodnocením.

Výraz je dosuptný pokud na každé cestě z Entry do P je E vypočtený a hodnoty nejsou následně změneny.

Výpočet dostupnosti je dopředný dataflow problém nad výrazy: výraz začne být dostupný v místě výpočtu. Přestává být dostupný pokud se změní některý z jeho parametrů. Sloučení dvou cest se provede pomocí operace and.

Propagace konstant nad celou funkcí

Definition (Plná redundance)

Výskyt výrazu E na pozici P je redundantní pokud je dostupný (available) před jeho vyhodnocením.

Výraz je dosuptný pokud na každé cestě z Entry do P je E vypočtený a hodnoty nejsou následně změneny.

Výpočet dostupnosti je dopředný dataflow problém nad výrazy: výraz začne být dostupný v místě výpočtu. Přestává být dostupný pokud se změní některý z jeho parametrů. Sloučení dvou cest se provede pomocí operace and.

Definition (Částečná redundance)

Výskyt výrazu E je částečně redundantní pokud je částečně dostupný (partially available) před jeho vyhodnocením.

Výraz je částečně dosuptný pokud na některé cestě z Entry do P je E vypočtený a hodnoty nejsou následně změneny.

Výpočet částečné dostupnosti je obdobné dataflow, jen sloučení se provádí operaci or (svaz se tím převrátí).

Propagace konstant nad celou funkcí

Definition (Plná redundance)

Výskyt výrazu E na pozici P je redundantní pokud je dostupný (available) před jeho vyhodnocením.

Výraz je dosuptný pokud na každé cestě z Entry do P je E vypočtený a hodnoty nejsou následně změneny.

Výpočet dostupnosti je dopředný dataflow problém nad výrazy: výraz začne být dostupný v místě výpočtu. Přestává být dostupný pokud se změní některý z jeho parametrů. Sloučení dvou cest se provede pomocí operace and.

Definition (Částečná redundance)

Výskyt výrazu E je částečně redundantní pokud je částečně dostupný (partially available) před jeho vyhodnocením.

Výraz je částečně dosuptný pokud na některé cestě z Entry do P je E vypočtený a hodnoty nejsou následně změneny.

Výpočet částečné dostupnosti je obdobné dataflow, jen sloučení se provádí operaci or (svaz se tím převrátí).

Partial redundancy elimination

GCSE

Globální eliinace redundatních podvýrazu (gloal common subexpression elimination, GCSE) je optimalize, která identifkuje redundatní podvýrazy (pomoci availability dataflow) a ty následně smaže.

Partial redundancy elimination

GCSE

Globální eliinace redundatních podvýrazu (gloal common subexpression elimination, GCSE) je optimalize, která identifkuje redundatní podvýrazy (pomoci availability dataflow) a ty následně smaže.

PRE

Eliminace částečně redundatních podvýrazů (partial redudancy ellimination, PRE) je optimalizace, která napřed vkládá nové výpočty daného výrazu E tak aby všechny částečně redundantní výpočty byly plně redundantní (code motion) a pak maže plně redundantní výrazy (GCSE)

PRE zobecňuje tradiční optimlizaci eliminující invarianty smyče (loop invariant motion).

Partially available expressions

Partial availability

Dopředný dataflow problém pro daný výraz E:

- V daném basic bloku:
  - Výraz je lokálně dostupný (AVLOC) pokud ho daný basic block počítá a potom už nemění hodnoty jeho parametrů.
  - Výraz umírá (KILL) pokud daný basic block mění hodnoty jeho parametrů.

Partially available expressions

Partial availability

Dopředný dataflow problém pro daný výraz E:

- V daném basic bloku:
  - Výraz je lokálně dostupný (AVLOC) pokud ho daný basic block počítá a potom už nemění hodnoty jeho parametrů.
  - Výraz umírá (KILL) pokud daný basic block mění hodnoty jeho parametrů.
- Polosvaz $\{0, 1\}$, sloučení $\cup$, $\top = 0$, $\mathit{Entry} = 0$

Partially available expressions

Partial availability

Dopředný dataflow problém pro daný výraz E:

- V daném basic bloku:
  - Výraz je lokálně dostupný (AVLOC) pokud ho daný basic block počítá a potom už nemění hodnoty jeho parametrů.
  - Výraz umírá (KILL) pokud daný basic block mění hodnoty jeho parametrů.
- Polosvaz $\{0, 1\}$, sloučení $\cup$, $\top = 0$, $Entry = 0$
- $\text{PAVOUT}[bb] = (\text{PAVIN}[bb] - \text{KILL}[bb]) \cup \text{AVLOC}[bb]$
- $\text{PAVIN}[bb] = \begin{cases} 0 & bb = \text{Entry} \\ \cup_{p \in \text{preds}(bb)} \text{PAVOUT}[p] & jinak \end{cases}$

Příklad

Code motion

Chci dosáhnout následujících vlastností:

Bezpečnost

Na žádnou cestu nevložím výpočet výrazu, ktery by už stejně nepočítala

Nově vložené výpočty mohou zpomalit program, ale taky vyhodit výjímku.

Code motion

Chci dosáhnout následujících vlastností:

Bezpečnost

Na žádnou cestu nevložím výpočet výrazu, ktery by už stejně nepočítala

Nově vložené výpočty mohou zpomalit program, ale taky vyhodit výjímku.

Optimaliza

Žádná cesta by neměla obsahovat redundantní výpočet

Bezpečnost

Očekávané výrazy (Anticipated expressions)

Zpětný dataflow problém pro daný výraz E:

- V daném basic bloku:
  - Výraz je lokálně očekávaný (AVLOC) pokud ho daný basic block počítá a před tím nemění hodnoty jeho parametrů.
  - Výraz umírá (KILL) pokud daný basic block mění hodnoty jeho parametrů.

Bezpečnost

Očekávané výrazy (Anticipated expressions)

Zpětný dataflow problém pro daný výraz E:

- V daném basic bloku:
  - Výraz je lokálně očekávaný (AVLOC) pokud ho daný basic block počítá a před tím nemění hodnoty jeho parametrů.
  - Výraz umírá (KILL) pokud daný basic block mění hodnoty jeho parametrů.
- Polosvaz $\{0, 1\}$, sloučení $\cap$, $\top = 1$, $\mathit{Exit} = 1$

Bezpečnost

Očekávané výrazy (Anticipated expressions)

Zpětný dataflow problém pro daný výraz E:

- V daném basic bloku:
  - Výraz je lokálně očekávaný (AVLOC) pokud ho daný basic block počítá a před tím nemění hodnoty jeho parametrů.
  - Výraz umírá (KILL) pokud daný basic block mění hodnoty jeho parametrů.
- Polosvaz $\{0, 1\}$, sloučení $\cap$, $\top = 1$, $Exit = 1$
- $\text{ANTIN}[bb] = (\text{ANTOUT}[bb] - \text{KILL}[bb]) \cup \text{ANTLOC}[bb]$
- $\text{ANDOUT}[bb] = \begin{cases} 0 & bb = \text{Exit} \\ \cap_{s \in \text{succs}(bb)} \text{ANTIN}[s] & jinak \end{cases}$

Kam vkádat výrazy?

Kam vkádat výrazy?

Morel-Renvoise 1979: Placement possible

Počítáme:

1. PPIN[*bb*] = můžeme vložit *E* do *bb* nebo dříve
2. PPOUT[*bb*] = můžeme vložit *E* do *bb* nebo později

Pak vložit na první pozici kde $PP = 1$

- INSERT[*bb*] $=$ PPOUT[*bb*] $\cap$ ($\neg$PPIN[*bb*] $\cup$ KILL[*bb*]) $\cap \neg AVOUT$[*bb*]
  (vkládáme jen na konec basic bloků)
- DELETE[*bb*] $=$ PPIN[*bb*] $\cap$ ANTLOC[*bb*]
  (mažeme jen první výskyty v basic bloku pokud před nimi se hodnota parametrů nezměnila)

Dvousměrné dataflow

PPOUT

$$\text{PPOUT}[bb] = \begin{cases} 0 & bb = \text{Exit} \\ \bigcap_{s \in \text{succs}(bb)} \text{PPIN}[s] & \textit{jinak} \end{cases}$$

(chceme vkládat na konci bloku pokud chceme vkládat na začátek všech potomků)

Dvousměrné dataflow

PPOUT

$$\text{PPOUT}[bb] = \begin{cases} 0 & bb = \text{Exit} \\ \bigcap_{s \in \text{succs}(bb)} \text{PPIN}[s] & \textit{jinak} \end{cases}$$

(chceme vkládat na konci bloku pokud chceme vkládat na začátek všech potomků)

PPIN

$$\text{PPIN}[bb] = \begin{cases} 0 & bb = \text{Exit} \\ ([\text{ANTLOC}[bb] \cup (\text{PPOUT}[bb] - \text{KILL}[bb])] & \\ \cap \bigcap_{p \in \text{preds}(bb)} (\text{PPOUT}[p] - \text{AVOUT}[p]) & \textit{jinak} \\ \cap \text{PAVIN}[bb]) & \end{cases}$$

Chceme vkládat na začátku bloku pokud máme lokální výpočet a nebo můžeme posunout výpočet z konce bloku na začátek.
Chceme posouvat výpočty všech předchůdců které nejsou ještě dostupné
Chceme posouváním něco získat (PAVIN jako heuristika)

Busy code motion

Knoop, Rüthing, Steffen v roce 1992 našli řešení pomocí jednosměrných dataflow problémů.

První pokus (busy code motion)

Výrazy vložíme na hranici anticipance: tedy na hrany z basic bloku kde výraz není očekávaný do basic blocku kde už očekávaný je.

Problém s oscilací

Busy code motion

Knoop, Rüthing, Steffen v roce 1992 našli řešení pomocí jednosměrných dataflow problémů.

První pokus (busy code motion)

Výrazy vložíme na hranici anticipance: tedy na hrany z basic bloku kde výraz není očekávaný do basic blocku kde už očekávaný je.

Problém s oscilací
Lépe: výpočet “will be available”, kde se tváříme, že basic block počítá výraz kdykoliv je očekáván.

(Will be) availability

Dopředný dataflow problém pro daný výraz E:

- Polosvaz $\{0, 1\}$, sloučení $\cap$, $\top = 1$, $\mathit{Entry} = 0$

Busy code motion

Knoop, Rüthing, Steffen v roce 1992 našli řešení pomocí jednosměrných dataflow problémů.

První pokus (busy code motion)

Výrazy vložíme na hranici anticipance: tedy na hrany z basic bloku kde výraz není očekávaný do basic blocku kde už očekávaný je.

Problém s oscilací
Lépe: výpočet “will be available”, kde se tváříme, že basic block počítá výraz kdykoliv je očekáván.

(Will be) availability

Dopředný dataflow problém pro daný výraz E:

- Polosvaz $\{0, 1\}$, sloučení $\cap$, $\top = 1$, $\mathit{Entry} = 0$
- $\mathrm{AVOUT}[bb] = (\mathrm{AVIN}[bb] - \mathrm{KILL}[bb]) \cup \mathrm{ANTIN}[bb]$
- $\mathrm{AVIN}[bb] = \begin{cases} 0 & bb = \mathrm{Entry} \\ \cap_{p \in \mathrm{preds}(bb)} \mathrm{AVOUT}[p] & \mathit{jinak} \end{cases}$

Algoritmus

$$\text{EARLIEST}(bb) = \text{ANTIN}[bb] - \text{AVIN}[bb]$$

Busy code motion

Pro každý výraz E vytvoř dočasnou proměnnou t:

- Pro každý basic block bb:
  - Nahraď všechny výpočty E za t.
  - Pokud $\text{EARLIEST}(bb) = 1$ vlož $t \leftarrow E$ na začátek bloku