Werkcollege Compilerconstructie
Vrijdag 6 december 2019

1. (a) Wat zijn *reaching definitions* op een bepaald punt in een programma?

   De algemene opzet van een iteratief algoritme van voorwaartse *dataflow* analyse is als volgt (waarbij de knopen de *basic blocks* zijn):

   OUT[ENTRY] = ...
   for each node B other than ENTRY
   OUT[B] = ...

   while (changes to any OUT occur)
   for each node B other than ENTRY
   { IN[B] = $\ldots_{\text{predecessors } P \text{ of } B}$ OUT[P]
   (i.e., combine the OUT-sets of the predecessors in some way)

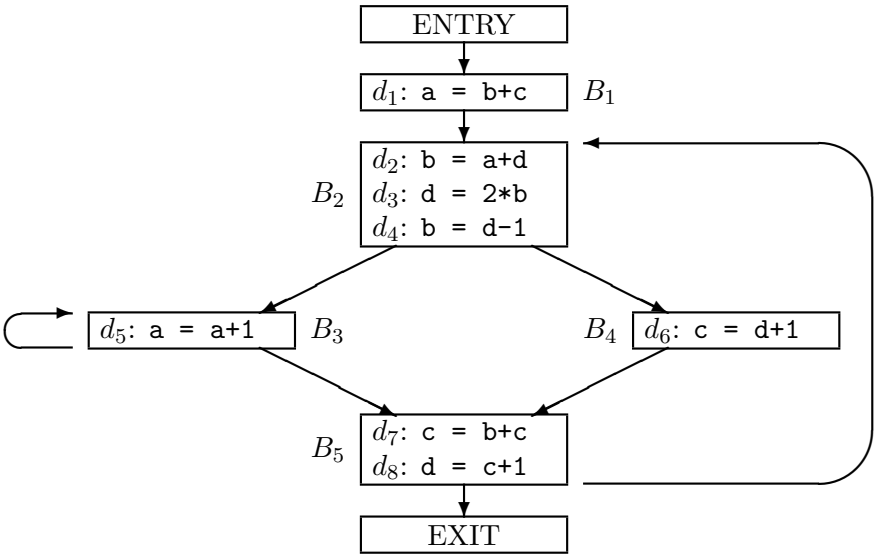
   OUT[B] = ... (some function of IN[B])
   }

   (b) Leg uit waarom we een *voorwaartse* (dus niet een achterwaartse) dataflow analyse gebruiken voor het berekenen van reaching definitions in een stroomdiagram (*flow graph*).

   (c) Vul de vier '...' in de algemene opzet van het iteratieve algoritme in voor het berekenen van reaching definitions. Voor iedere knoop B in het stroomdiagram moeten IN[B] en OUT[B] aan het eind van het algoritme alle reaching definitions aan het begin, respectievelijk einde van B bevatten.

   Wees met name ook precies in je beschrijving van 'some function'.

Beschouw nu het volgende stroomdiagram:

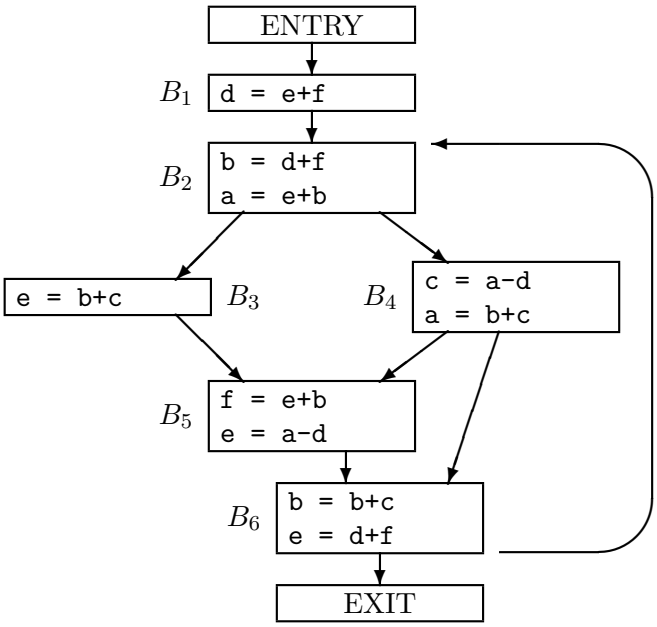


(d) Wanneer we het iteratieve algoritme willen gebruiken om de reaching definitions in een stroomdiagram te berekenen, moeten we een volgorde kiezen waarin we de basic blocks aflopen in de binnenste for-lus (dwz: de for-lus binnen de while-lus in het algoritme).

Wat is een gunstige volgorde van de basic blocks bij bovenstaand stroomdiagram? Motiveer je antwoord.

(e) Pas nu het iteratieve algoritme om de reaching definitions te berekenen toe op bovenstaand stroomdiagram. Laat met een overzichtelijke tabel zien wat de IN en OUT-verzameling van elk basic block (op ENTRY na) is na de initialisatie en na iedere iteratie van de while-lus. De laatste iteratie, waarin je zou constateren dat er toch niets veranderd is, hoef je niet uit te voeren.

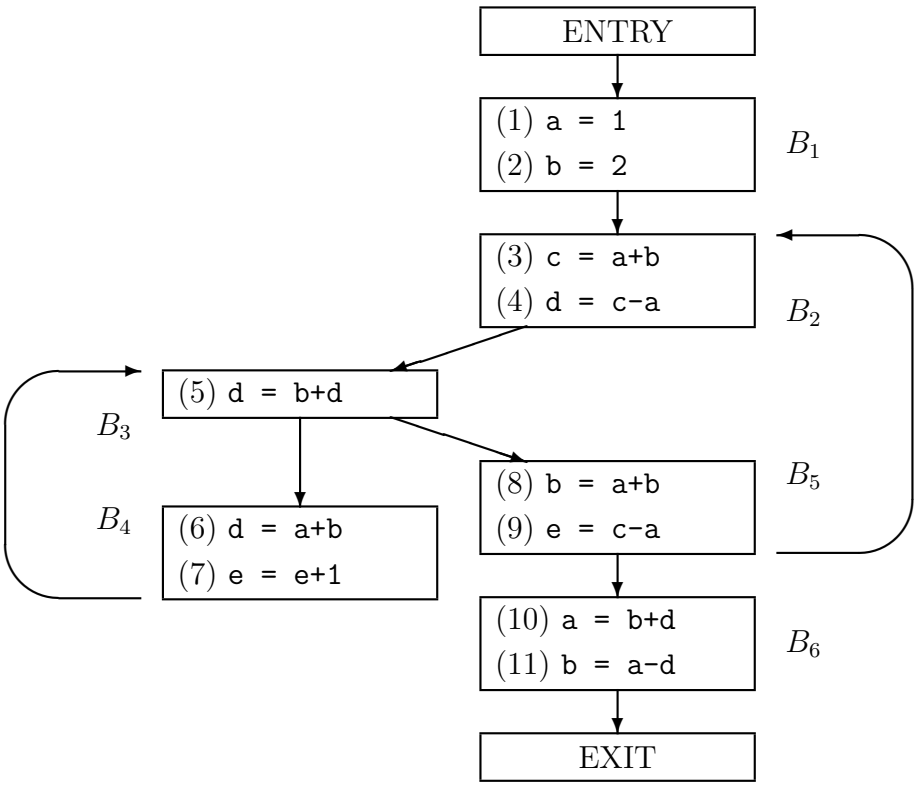
2. Deze opgave gaat over de dataflow analyse om available expressions te bepalen. Beschouw het volgende stroomdiagram:



(a) Bepaal voor elk blok $B \in \{B_1, B_2, \ldots B_6\}$ zowel e_gen_B als e_kill_B.

(b) Pas het iteratieve algoritme om de available expressions te berekenen toe op bovenstaand stroomdiagram. Laat met een overzichtelijke tabel zien wat de IN en OUT-verzameling van elk basic block (op ENTRY na) is na de initialisatie en na iedere iteratie van de while-lus. De laatste iteratie, waarin je zou constateren dat er toch niets veranderd is, hoef je niet uit te voeren.

Consider the following flow graph:



Exercise 9.2.1. Reaching definitions.
For the above flow graph, compute

(a) The *gen* and *kill* sets for each block.

(b) The IN and OUT sets for each block.

Exercise 9.2.2. Available expressions.
For the above flow graph, compute

(a) The *e_gen* and *e_kill* sets for each block.

(b) The IN and OUT sets for each block.

Exercise 9.2.3. Live variables.
For the above flow graph, compute

(a) The *def* and *use* sets for each block.

(b) The IN and OUT sets for each block.