Dead Argument elimination

Filip Gajin 119/2020

Optimizacija Dead Argument Elimination

- Dead Argument Elimination je optimizaciona tehnika u kompilatorima koja uklanja argumente funkcija koji se nikada ne koriste. Ova transformacija smanjuje broj parametara u funkcijama, ubrzava pozive i pojednostavljuje kontrolni tok programa.
- Ideja je jednostavna:
- Ako funkcija ima argument koji nema nijednu “pravu” upotrebu u telu funkcije (tj. ne utiče na rezultat ili kontrolu), taj argument se može bezbedno ukloniti, zajedno sa svim pozivima koji su ga prosleđivali.
- Ovo je tip interproceduralne optimizacije, jer može uticati i na druge funkcije koje pozivaju datu.
- DAE se obično koristi kao "cleanup" pass nakon drugih optimizacija (npr. inlining, constant propagation, DCE), da bi se uklonili suvišni argumenti koji su postali mrtvi nakon što su izrazi ili pozivi eliminisani.

Algoritam

Analiza funkcije – proverava se za svaki argument da li ima “realne” upotrebe (tj. da li se koristi u instrukcijama koje nisu llvm.dbg.*)
Označavanje mrtvih argumenata – oni koji se ne koriste ni direktno ni indirektno (putem drugih funkcija) označavaju se kao “dead”.
Kloniranje funkcije – pravi se nova verzija funkcije bez mrtvih argumenata, a stara se uklanja.
Prevezivanje poziva – svi call i invoke pozivi se prepravljaju da pozivaju novu funkciju sa filtriranim listama argumenata

Implementacija

- **Detekcija mrtvih argumenata**

Svaki argument se proverava pomoću funkcije hasNonDebugUses(). Ako argument nema nijednu upotrebu osim DbgInfoIntrinsic, smatra se mrtvim.

- **Kloniranje funkcije**

Funkcija se klonira pomoću CloneFunctionInto, ali uz pažljivo kreiran novi FunctionType koji sadrži samo “žive” argumente. Za mrtve argumente u ValueToValueMap stavljamo UndefValue::get(A.getType()), čime se izbegava problem sa context mismatchom.

- **Ažuriranje svih poziva**

Svaki CallInst i InvokeInst koji poziva staru funkciju se prepravlja tako da poziva novu funkciju sa istim “živim” argumentima. Stara funkcija se zatim briše (eraseFromParent()), a nova dobija originalno ime.

- **Verifikacija ispravnosti**

Nakon svake transformacije poziva se verifyFunction() da bi se proverilo da nije došlo do narušavanja LLVM IR konzistentnosti.

Testovi

- Jednostavna funkcija sa mrtvim argumentom
- Funkcije koje pozivaju jedna drugu
- Lancani primer
- Negativan test u kom ne treba da promeni nista

- Svi testovi su uspesno prosli