Data Flow Analysis

There are two types of analysis performed for global optimizat (optimations across basic blocks) namely Control flow analysis and data flow analysis.

Control flow analysis : Optimization is performed based on the analytical information regarding arrangement of basic block loops and nodes. Thus analysis is made on the flow of control by examining the flow graph.

Data flow analysis : Analysis here is made on the flow o data. Optimization is done based on the information regarding the definition and use of the data in the program. The values are computed using data flow proper on Data Flow schemas on basic blocks.

Data flow properties :— Data flow analysis is a process of (or schemas) computing values of data flow prope

1. Available expressions : An expression x+y is available at a program point w if and only if along all paths it is reaching to w. It is used to eliminate comm subexpressions.

2. Reaching definitions : A definition D reaches a point if there is a path from D to p along which D is not killed. A definition D of variable x is killed when there is a redefinition of x. These are used in constant and variable propagations.

3) Live Variables: A variable x is live at some point if there is a path from p to the exit, along which value of x is used before it is redefined. Otherwise the variable is said to be dead at that point. They are useful in register allocation and dead code elimination.

4) Busy Expressions: An expression 'e' is said to be a busy expression along path $p_i \cdots p_j$ if and only if an evaluation of 'e' exists along $p_i \cdots p_j$ and no definition of any operand exists before its evaluation along the path. These are useful in performing code movement optimizations.

Data Flow Equation: The information an optimizing compiler collects by data-flow-analysis is called data-flow information. This can be collected by setting up and solving equations that relate information at various points in a program.

$$Out[s] = gen[s] \cup (in[s] - kill[s])$$

The above equation is read as "the information at the end of a statement is either generated within the statement or enters at the beginning and is not killed as control flows through the statement."

Such equations are called dataflow equations, where gen[s] — set of definitions generated by statement-s, kill[s] — is

A dataflow analysis framework $(D, V, \wedge, F)$ consists of

1. A direction of data flow D, which is either FORWARDS or BACKWARDS.
2. A semilattice, which includes domain of values V and a meet operator $\wedge$.
3. A family F of transfer functions from V to V, which include ENTRY and EXIT (boundary conditions) in any flow graph.

Semilattices: A semilattice is a set V and a binary meet operator $\wedge$ such that $\forall x, y, z$ in V:

1. $x \wedge x = x$ (meet is idempotent)
2. $x \wedge y = y \wedge x$ (meet is commutative)
3. $x \wedge (y \wedge z) = (x \wedge y) \wedge z$ (meet is associative)

A semilattice has a top element T, such that

$\forall x$ in V, $T \wedge x = x$

A semilattice may have a bottom element $\perp$, such that $\forall x$ in V, $\perp \wedge x = \perp$