Program Analysis

Data Flow Analysis (Part 1)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Winter 2023/2024

Big Picture

- Static versus dynamic analysis
- Many ways of formulating and implementing analyses
- One popular way of formulating a static analysis: Data flow analysis

Real-World Use Cases

Many IDE features are based on data flow analysis

- **E.g.**
  - Reaching definitions
  - Unused variables

Data Flow Analysis

Basic idea

- Propagate analysis information along the edges of a control flow graph
- Goal: Compute analysis state at each program point
- For each statement, define how it affects the analysis state
- For loops: Iterate until fix-point reached

Outline

- First example: Available expressions ←
- Basic principles
- More examples
- Solving data flow problems
- Inter-procedural analysis
- Sensitivities

Available Expression Analysis

Goal: For each program point, compute which expressions must have already been computed, and not later modified

- Useful, e.g., to avoid re-computing an expression
- Used as part of compiler optimizations

Example

```
var x = a + b;
var y = a * b;
while (y > a + b) {
  a = a - 1;
  x = a + b;
}
```

Example

```
var x = a + b;
var y = a * b;
while (y > a + b) {
  a = a - 1;
  x = a + b;
}
```

Available every time execution reaches this point

Transfer Functions

- **Transfer function of a statement:**
  How the statement affects the analysis state
  - Here: Analysis state = available expressions
- **Two functions**
  - gen: Available expressions generated by a statement
  - kill: Available expressions killed by a statement

gen Function

Function $gen : Stmt \rightarrow \mathcal{P}(Expr)$

- A statement generates an available expressions e if
  - it evaluates e and
  - it does not later write any variable used in e
- Otherwise, function returns empty set

Example:

`var x = a * b;` generates `a * b`

kill Function

Function $kill : Stmt \rightarrow \mathcal{P}(Expr)$

- A statement kills an available expressions e if
  - it modifies any of the variables used in e
- Otherwise, function returns empty set

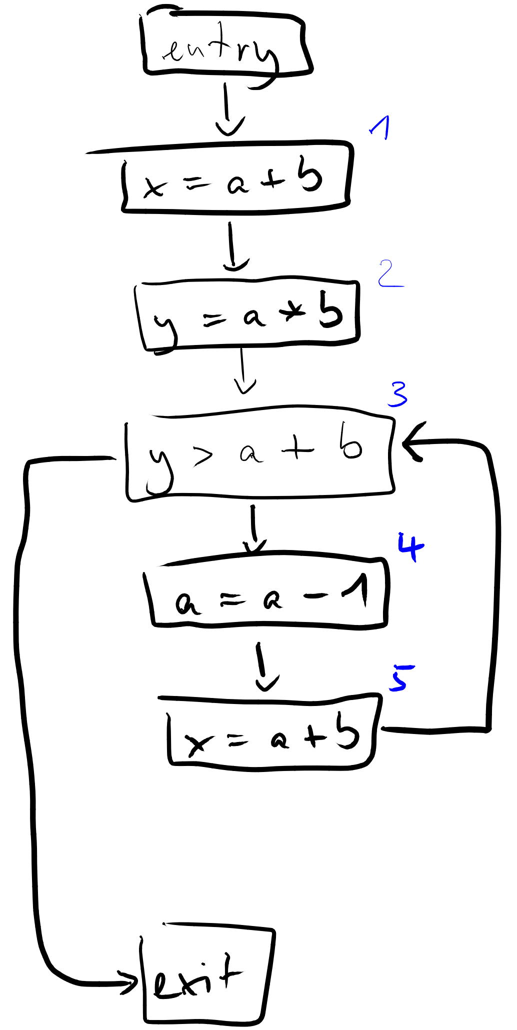
Example:

`a = 23;` kills `a * b`

Example

```
var x = a + b;
var y = a * b;
while (y > a + b) {
  a = a - 1;
  x = a + b;
}
```

Control flow graph



Non-trivial expressions:

$a + b$

$a * b$

$a - 1$

Transfer function for each statement

Statement s	gen(s)	kill(s)
1	$\{a + b\}$	$\emptyset$
2	$\{a * b\}$	$\emptyset$
3	$\{a + b\}$	$\emptyset$
4	$\emptyset$	$\{a - 1, a + b, a * b\}$
5	$\{a + b\}$	$\emptyset$

Propagating Available Expressions

- Initially, no available expressions
- Forward analysis: Propagate available expressions in the direction of control flow
- For each statement s, outgoing available expressions are:
  incoming avail. exprs. minus $kill(s)$ plus $gen(s)$
- When control flow splits, propagate available expressions both ways
- When control flows merge, intersect the incoming available expressions