Fast Liveness Checking for SSA-Form Programs

Benoit Boissinot
Equipe Compsys
E.N.S Lyon

October 18, 2007

Outline

Outline

Optimizing compilers

- High level languages, retargetable compiler
- Different interdependant passes

Optimizing compilers

- High level languages, retargetable compiler
- Different interdependant passes

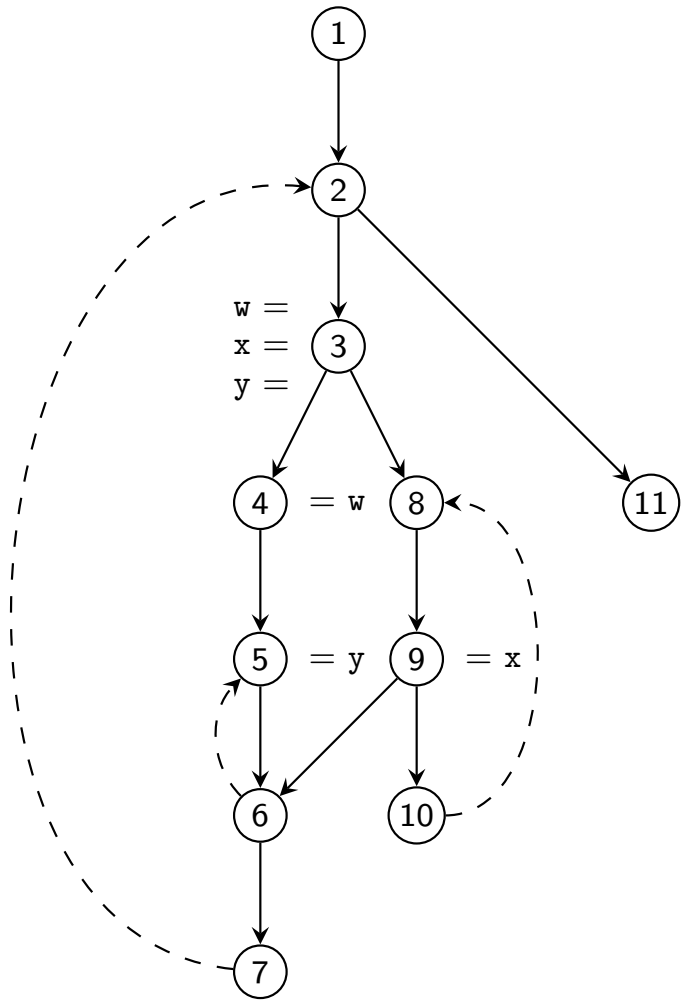
Compile time / JIT

- Run everywhere
- Dynamic optimization

Outline

Control Flow Graph

- $G = (V, E, r)$ directed graph
- $r \in V$ is the root
- nodes are basic blocks



Dominance

Definition (Dominance)

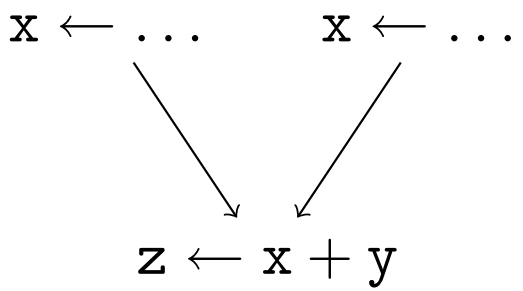
A node x in a control-flow graph dominates another node y ($x \operatorname{dom} y$) if every path from r to y contains x. A dominance relation is said to be strict if $x \neq y$.

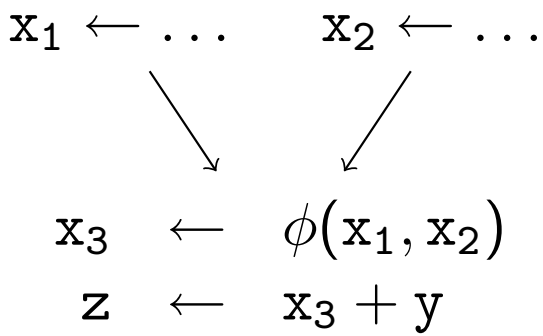
Static Single Assignment (SSA)

For each variable:

- Only one definition
- Each use is dominated by the definition (strictness)

At control flow merge point, the ϕ-function acts as a switch.





Liveness

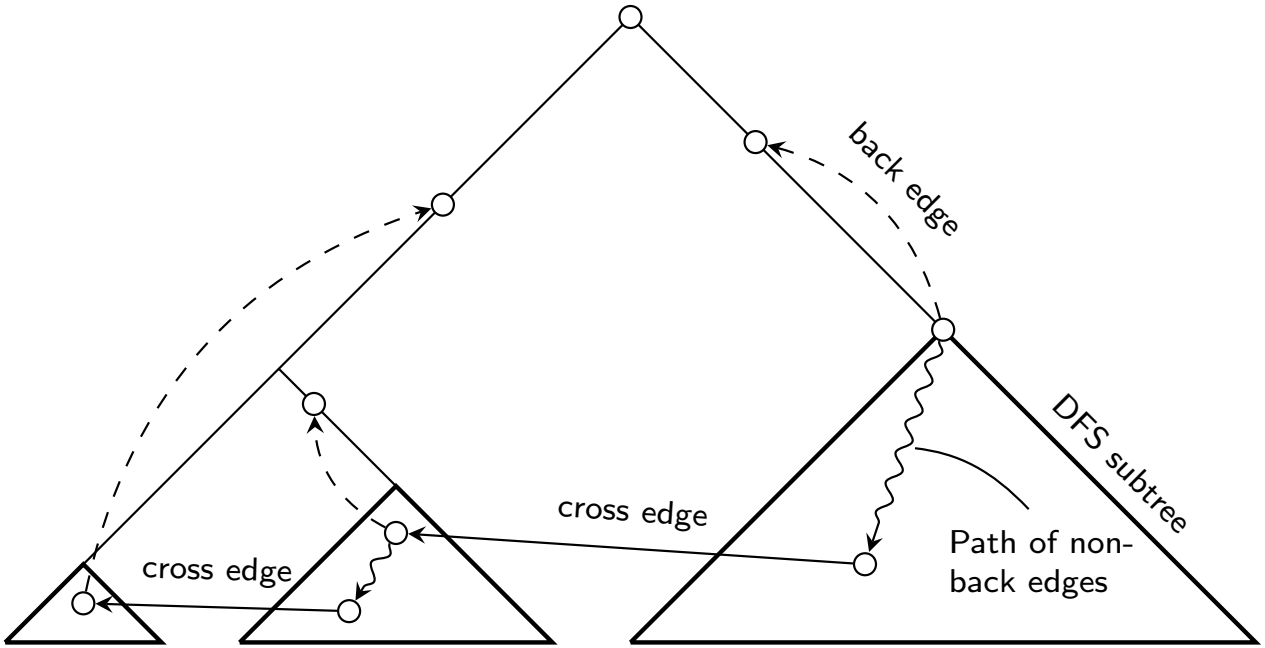
Definition (live-in)

A variable a is live-in at a node q if there exists a path from q to a node u where a is used and that path does not contain def(a).

Liveness analysis

- Two passes:
  - precomputation (data-flow)
  - query (live-in sets)

Depth First Search



Reduced Graph

Back edges

$$E^{\uparrow} = \{(s,t) \in E \mid t \text{ is an ancestor of } s\}$$

Reduced Graph

$$\widetilde{G} = (V, E \setminus E^{\uparrow}, r)$$

Reduced Reachability

$$R_v = \{w \in V \mid \exists \text{ path } v \rightarrow w \text{ in } \widetilde{G}\}$$

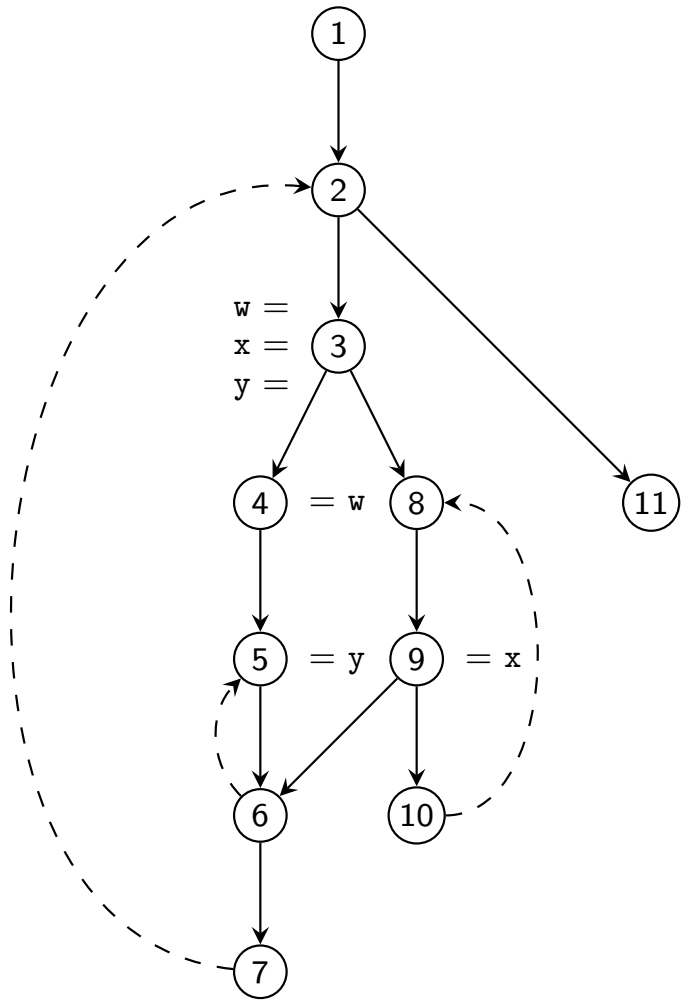
Concepts

Idea

Break the path into back-edge-free subpaths and ensure they don't go through the definition.

Simple Path

If a path contains no backedges, simple:
def(a) sdom q and $u \in R_q$?



Concepts

Main Principle

- T_q: set of back edges target reduced reachable from q
- a live-in at q is equivalent to $\exists t \in T_q, def(\mathrm{a})\ \mathrm{sdom}\ t$ and $u \in R_t$

Concepts

Main Principle

- T_q: set of back edges target reduced reachable from q
- a live-in at q is equivalent to $\exists t \in T_q, def(\mathrm{a})\ \mathrm{sdom}\ t$ and $u \in R_t$
- The path to the back edge should not re-enter the dominance subtree

Concepts

Definition (T_q)

$$
\begin{aligned}
T_t^{\uparrow} &= \left\{ t' \in V \setminus R_t \mid \exists s' \in R_t \wedge (s', t') \in E^{\uparrow} \right\} \\
T_q^0 &= \{q\} \\
T_q^i &= \bigcup_{t \in T_q^{i-1}} T_t^{\uparrow} \\
T_q &= \bigcup_{i=0}^{\infty} T_q^i
\end{aligned}
$$

Algorithm

Query Algorithm

1: **function** IsLiveIn(variable a, node q)
2: $T_{(q,\mathrm{a})} \leftarrow T_q \cap sdom(def(\mathrm{a}))$
3: **for** $t \in T_{(q,\mathrm{a})}$ **do**
4: **if** $R_t \cap uses(\mathrm{a}) \neq \emptyset$ **then** **return** *true*
5: **return** *false*

Algorithm

Query Algorithm

1: **function** IsLiveIn(variable a, node q)
2: $T_{(q,\mathrm{a})} \leftarrow T_q \cap sdom(def(\mathrm{a}))$
3: **for** $t \in T_{(q,\mathrm{a})}$ **do**
4: **if** $R_t \cap uses(\mathrm{a}) \neq \emptyset$ **then** **return** *true*
5: **return** *false*

Further Details

- use bitsets for T_q
- using the dominance order minimizes the number of query

Outline

Implementation

- Implemented in LAO, code generator developped by STMicroelectronics

Implementation

- Implemented in LAO, code generator developped by STMicroelectronics
- Benchmarked with a subset of SPEC2000 (CINT)

Implementation

- Implemented in LAO, code generator developped by STMicroelectronics
- Benchmarked with a subset of SPEC2000 (CINT)
- Liveness-analysis used during SSA deconstruction

Quantitative Evaluation

The main factors influencing the speed of our algorithm are:

Quantitative Evaluation

The main factors influencing the speed of our algorithm are:

- the length of the def-use chain; used in the for loop of the query algorithm.

Quantitative Evaluation

The main factors influencing the speed of our algorithm are:

- the length of the def-use chain; used in the for loop of the query algorithm.
- the number of basic blocks since it determines the size of the bitsets T_v and R_v.

Quantitative Evaluation

The main factors influencing the speed of our algorithm are:

- the length of the def-use chain; used in the for loop of the query algorithm.
- the number of basic blocks since it determines the size of the bitsets T_v and R_v.
- the number of CFG edges since they govern the time to precompute the T_v and R_v.

Quantitative Evaluation

	# of Uses per Variable				
Benchmark	Maximum	% ≤ 1	% ≤ 2	% ≤ 3	% ≤ 4
164.gzip	51	65.64	86.38	92.81	95.94
175.vpr	75	70.36	88.90	93.93	96.28
176.gcc	422	73.99	87.81	92.42	94.84
181.mcf	46	66.91	83.50	89.33	94.46
186.crafty	620	72.98	90.09	93.85	95.75
197.parser	96	65.12	86.75	94.26	96.62
254.gap	156	70.46	85.95	91.26	94.54
255.vortex	254	65.99	90.80	95.02	96.97
256.bzip2	36	69.89	89.89	94.47	96.17
300.twolf	165	69.71	87.59	93.23	95.92
Total	620	71.30	87.85	92.76	95.31

Quantitative Evaluation

	# of Basic Blocks			
Benchmark	Average	Sum	% $\leq$ 32	% $\leq$ 64
164.gzip	33.35	2735	69.51	85.36
175.vpr	34.45	7752	68.88	84.44
176.gcc	38.96	78666	72.85	86.03
181.mcf	20.31	528	84.61	100.00
186.crafty	69.28	7551	59.63	76.14
197.parser	23.60	7623	84.82	93.49
254.gap	32.89	28020	67.60	87.44
255.vortex	26.46	24425	77.57	90.68
256.bzip2	22.97	1700	78.37	91.89
300.twolf	56.97	10825	59.47	77.36
Total	35.21	169825	72.71	87.18

Runtime Experiments

	Precomputation		Queries		Both
Benchmark	# Proc.	Spdup	# Queries	Spdup	Spdup
164.gzip	82	3.12	90659	0.53	1.16
175.vpr	225	2.17	55670	0.48	1.41
176.gcc	2019	3.03	1109202	0.26	1.00
181.mcf	26	1.85	2369	0.44	1.39
186.crafty	109	2.78	858121	0.49	0.73
197.parser	323	2.13	38719	0.49	1.54
254.gap	852	3.45	245540	0.52	2.08
255.vortex	923	1.67	88554	0.45	1.32
256.bzip2	74	3.45	10100	0.51	2.32
300.twolf	190	4.76	184621	0.49	1.92
Total	4823	2.94	2683555	0.36	1.16

Outline

Contributions

- Novel approach for liveness checking relying only on the CFG
- Fast construction algorithm
- Overall speedup in most cases

Future Work

- Memory efficient transitive closure
- Dynamic update for CFG transformations
- Use information available from the loop nesting forest

The End

Thank you!