Points-to Analysis with Efficient Strong Updates

Ondřej Lhoták
K.-C. Andrew Chung
Points-to analysis

\[
\begin{align*}
a &= 1; \\
b &= 2; \\
\ast x &= 4; \\
c &= a + b;
\end{align*}
\]

If \( x == &a \), then \( c = 6 \).
If \( x == &b \), then \( c = 5 \).
If \( x != &a && x != &b \), then \( c = 3 \).
Subset-based points-to analysis

\[ p = &a; \]
\[ q = &b; \]
\[ r = &c; \]
\[ p = q; \]
\[ q = p; \]
\[ r = q; \]
Flow-sensitive points-to analysis

\[ p = \&a; \]
\[ p \rightarrow a \]
\[ q = \&b; \]
\[ p \rightarrow a \quad q \rightarrow b \]
\[ r = \&c; \]
\[ p \rightarrow a \quad q \rightarrow b \quad r \rightarrow c \]
\[ p = q; \]
\[ p \rightarrow b \quad q \rightarrow b \quad r \rightarrow c \]
\[ q = p; \]
\[ p \rightarrow b \quad q \rightarrow b \quad r \rightarrow c \]
\[ r = q; \]
\[ p \rightarrow b \quad q \rightarrow b \quad r \rightarrow b \]

Strong updates

Flow-insensitive result
In SSA form, top-level pointers are easy

\[
p_1 = \&a; \\
p_1 \rightarrow a \\
q_1 = \&b; \\
q_1 \rightarrow b \\
r_1 = \&c; \\
r_1 \rightarrow c \\
p_2 = q_1; \\
p_2 \rightarrow b \\
q_2 = p_2; \\
q_2 \rightarrow b \\
r_2 = q_2; \\
r_2 \rightarrow b
\]
Goal

Can we compute a FS points-to analysis as efficiently as a FI one?
Can we compute an almost FS points-to analysis almost as efficiently as a FI one?
Challenge

\[
p = &a;
\]
\[
p \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ q \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ r \rightarrow \text{a}\ \text{b}\ \text{c}
\]
\[
q = &b;
\]
\[
p \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ q \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ r \rightarrow \text{a}\ \text{b}\ \text{c}
\]
\[
r = &c;
\]
\[
p \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ q \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ r \rightarrow \text{a}\ \text{b}\ \text{c}
\]
\[
p = q;
\]
\[
p \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ q \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ r \rightarrow \text{a}\ \text{b}\ \text{c}
\]
\[
q = p;
\]
\[
p \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ q \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ r \rightarrow \text{a}\ \text{b}\ \text{c}
\]
\[
r = q;
\]
\[
p \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ q \rightarrow \text{a}\ \text{b}\ \text{c}\ \ \ r \rightarrow \text{a}\ \text{b}\ \text{c}
\]

\[O(\text{AVL})\]
Previous work

Hardekopf & Lin 2009
- binary decision diagrams (BDDs)
- sparse SSA for top-level variables

Goyal 2005, Staiger-Stöhr 2009
- incremental computation of sparse dependence graph

Both compute *fully* FS analysis result.
We settle for *almost* FS result.
Key idea:

The specific precision required for strong updates turns out to be cheap.
Example: flow-sensitive analysis

Flow-insensitive result:

- `p` points to `a`
- `q` points to `b e`
- `r` points to `b e`
- `a` points to `b e`
- `b` points to `c d f`
- `e` points to `c d f`

```
p = &a;
*p = &b;
q = *p;
*a = b
*q = &c;
*b = c
if(*) *p = &e;
*r = *p;
*a = b e
*r = &f;
```

Top-level points-to sets:

- `p` points to `a`
- `q` points to `b`
- `r` points to `b e`

```
if(*) *p = &e;
```

- strong update

```
r = *p;
*a = b e
*r = &f;
```

- weak update
Design of Strong Update Analysis

• Do FI and FS analysis together at the same time.

• Track only singleton points-to sets precisely: FS.
  • cheap to do since they’re singletons

• Keep FI sets as backup for non-singletons.
Example: Strong Update Analysis

\[ p = \&a; \]
\[ *p = \&b; \]
\[ q = *p; \]
\[ *q = \&c; \]
\[ *q = \&d; \]
\[ \text{if}(*) \] \[ *p = \&e; \]
\[ r = *p; \]
\[ *r = \&f; \]

Flow-insensitive backup sets:

\[ p \rightarrow a \]
\[ q \rightarrow b \]
\[ r \rightarrow b e \]
\[ a \rightarrow b e \]
\[ b \rightarrow c d f \]
\[ e \rightarrow f \]
Comparison: FS vs. SU

\[ p = \&a; \]

\[ *p = \&b; \]
\[ q = *p; \]
\[ a \rightarrow b \]
\[ a \rightarrow b \]
\[ *q = \&c; \]
\[ a \rightarrow b \]
\[ b \rightarrow c \]

\[ \text{if}(*) \] \[ *p = \&e; \]
\[ a \rightarrow b \]
\[ b \rightarrow d \]
\[ r = *p; \]
\[ a \rightarrow b \]
\[ b \rightarrow d \]
\[ *r = \&f; \]
\[ a \rightarrow b \]
\[ b \rightarrow c \]
\[ c \rightarrow d \]
\[ d \rightarrow e \]
\[ e \rightarrow f \]

Flow-insensitive backup sets:

- \[ p \rightarrow a \]
- \[ q \rightarrow b \]
- \[ r \rightarrow b, e \]
- \[ a \rightarrow b, e \]
- \[ b \rightarrow c, d, f \]
- \[ e \rightarrow f \]

The only difference
- strong update
- weak update

Points-to Analysis with Efficient Strong Updates | Ondřej Lhoták, K.-C. Andrew Chung | University of Waterloo
Implementation simplicity

• We can extend any FI implementation with SU.

• Common implementation tricks can be reused (e.g. online cycle detection).

• Our implementation is in LLVM 2.6.

• Our evaluation is on SPEC CPU 2000 and 2006.
Experimental results: speed

![Analysis time (s) graph]

- 164.gzip
- 175.vpr
- 176.gcc
- 181.mcf
- 186.crafty
- 197.parser
- 253.perlbmk
- 254.gap
- 255.vortex
- 256.bzip2
- 300.twolf
- 400.perlbench
- 401.bzip2
- 403.gcc
- 429.mcf
- 433.milc
- 445.gobmk
- 456.hmmer
- 458.sjeng
- 462.libquantum
- 464.h264ref
- 470.lbm
- 482.sphinx3

- **FI**
- **SU**
Experimental results: precision

Number of Strong Updates

- 164.gzip
- 175.vpr
- 181.mcf
- 186.crafty
- 197.parser
- 254.gap
- 255.vortex
- 256.bzip2
- 300.twolf
- 401.bzip2
- 429.mcf
- 433.milc
- 445.gobmk
- 456.hmmer
- 458.sjeng
- 462.libquantum
- 464.h264ref
- 470.lbm
- 482.sphinx3

SU
FS
Total Stores

Points-to Analysis with Efficient Strong Updates | Ondřej Lhoták, K.-C. Andrew Chung | University of Waterloo
Experimental results: precision

Number of Loads More Precise than FI

1 10 100 1000 10000 100000

SU FS

Total Loads

Points-to Analysis with Efficient Strong Updates | Ondřej Lhoták, K.-C. Andrew Chung | University of Waterloo
Conclusions

• SU analysis is a **simple** extension of FI analysis.

• SU analysis is **almost** as **fast** as FI.
  • 9% faster to 22% slower (5% on average)

• SU analysis is **almost** as **precise** as FS.
  • 98% of strong updates
  • 98% of more precise loads