What is the VELOCITY Low-Level Pointer Analysis?

To overcome the limitations of high-level pointer analysis,
VELOCITY Low-Level Pointer Analysis (VLLPA) operates
at the assembly level. VLLPA is context sensitive and
partially flow-sensitive (tracks registers according to their
position in the CFG). It handles all the features of C, such as
unions. It is also modular in that only part of the program and
the analysis information needs to be present in memory
simultaneously, reducing the memory requirement of context-
sensitive interprocedural analysis

How accurate is VLLPA compared to high-level
analysis? The IMPACT compiler’s powerful high-level
pointer analysis is context-sensitive and flow-insensitive. A
comparison with IMPACT shows the number of dependence
arcs between memory operations computed from the pointer
analysis results. On average, VLLPA is less precise on only
0.3% of memory operations and more precise on 26.8% of
memory operations.

How bad is the problem of conservative memory
dependence propagation? Dependence arcs created by
the high-level analysis must be conservatively propagated
by subsequent code transformations, resulting in many spurious
dependences. VLLPA was ran on the low-level code
produced by IMPACT at the final stage of compilation to
produce accurate dependence arcs. The dependence arcs
propagated from IMPACT’s high-level pointer analysis to the
final stage were then classified in three categories:
1) Correct: Arcs identified by VLLPA.
2) Unnecessary: Spurious arcs resulting from the propagation
of high-level arcs that were themselves spurious.
3) Unnecessarily Propagated: Spurious arcs resulting from the
propagation of accurate high-level arcs.
The number in the third category accounts for up to 50%
of all IMPACT arcs present at code generation. This shows that
repeating memory analysis at the low level is significantly
more accurate than propagating memory analysis information
from the high level.

More information:
http://www.liberty-research.org/Research/VLLPA or contact
the Liberty Low-Level Pointer Analysis Team at the
addresses above. Bolei Guo, Matthew J. Bridges, Spyridon Triantafyllis, Guilherme Ottoni, Easwaran Raman, David I.
August, “Practical and Accurate Low-Level Pointer
Analysis”, Proceedings of the Third International Symposium

Accuracy Comparison with IMPACT: The third and fourth columns
show the number of memory operations for which VLLPA results in
fewer and more dependence arcs. Fewer arcs imply high precision.

Conservative Dependence Propagation Example: After superblock
formation is performed on the source code, instructions 2, 3 and 4
are grouped into a superblock (instructions 3’ and 4’ are created
for tail duplication). There is no longer any dependence between 3
and 4, but the superblock formation algorithm conservatively
propagates the dependence relation to the transformed code.

Breakdown of Propagated High-Level Dependence Arcs: See text.