Re-Engineering Abstract Interpretation

Inferring Contracts and Proving Program Properties

S. Tucker Taft, AdaCore *taft@adacore.com*

AdaCore The GNAT Pro Company

Overview

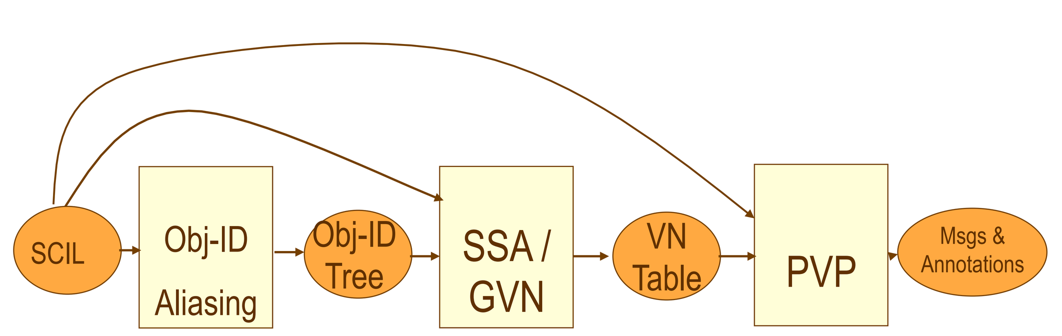
CodePeer is an advanced static analysis tool based on interprocedural, global, *control and data flow* analysis.

CodePeer *infers* contracts (pre/postconditions) and proves *absence* of run-time errors (AoRTE).

CodePeer is an example of *Abstract Interpretation*, but incorporates lessons learned from decades of *optimizing compiler* development, to produce *precise* and *sound* results very efficiently, based on a systematic, *scalable*, bottom-up approach.

Overall Structure

CodePeer structure

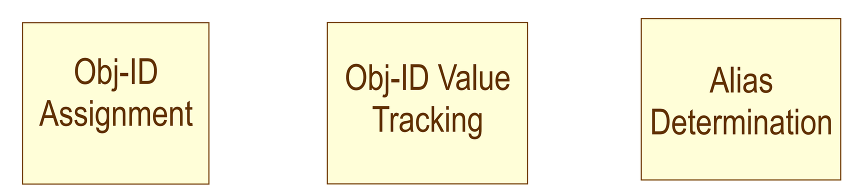


Kinds of Annotations Inferred:

- Inputs (Live-On-Entry)
- Outputs (DMODs – Direct Modifications)
- New Objects (Escape Analysis),
- Preconditions, Postconditions, Presumptions

Object Identification – Bottom-up Aliasing Analysis

Obj-ID



- Obj-ID assigned to each object "name"
- Extra assignments to represent parameter passing and "pseudo" fields of objects (e.g. taintedness)
- Values of pointer/integer Obj-IDs tracked (mostly flow insensitive in this phase), DMODs and exports id'ed
- Pointer and array index value sets used to determine potential aliases
- Value sets of DMODs/exports propagated to callers (values possibly assigned by callee added to caller set)

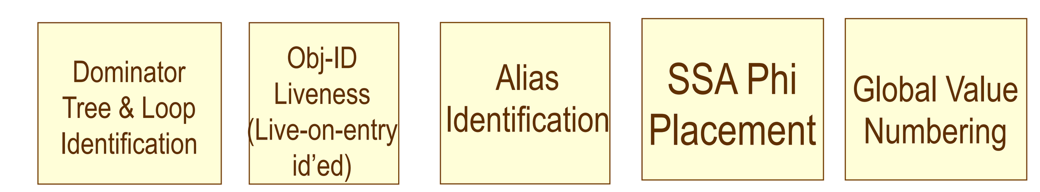
Vocabulary for Obj-ID, SSA/GVN, PVP

Vocabulary

- Module, Procedure, Init-Procedure
- Basic Block, Edge, Predecessor, Successor, Statement
- Obj-ID, DMOD, Export/New object, Live on Entry
- Static Single Assignment (SSA)
  - Phi and Kappa and Phis-as-Kappa
  - Initial-External-Kappa, Aliased-Kappa, Multi-Target-Call Kappa
- Value Numbers (VNs) and VN Computation Table
- Possible Value Set Propagation (PVP)
  - Constraint Propagation, Arc Consistency
- Unknown/Unanalyzed Module/Procedure/Call/Result

Static Single Assignment & Global Value Numbering

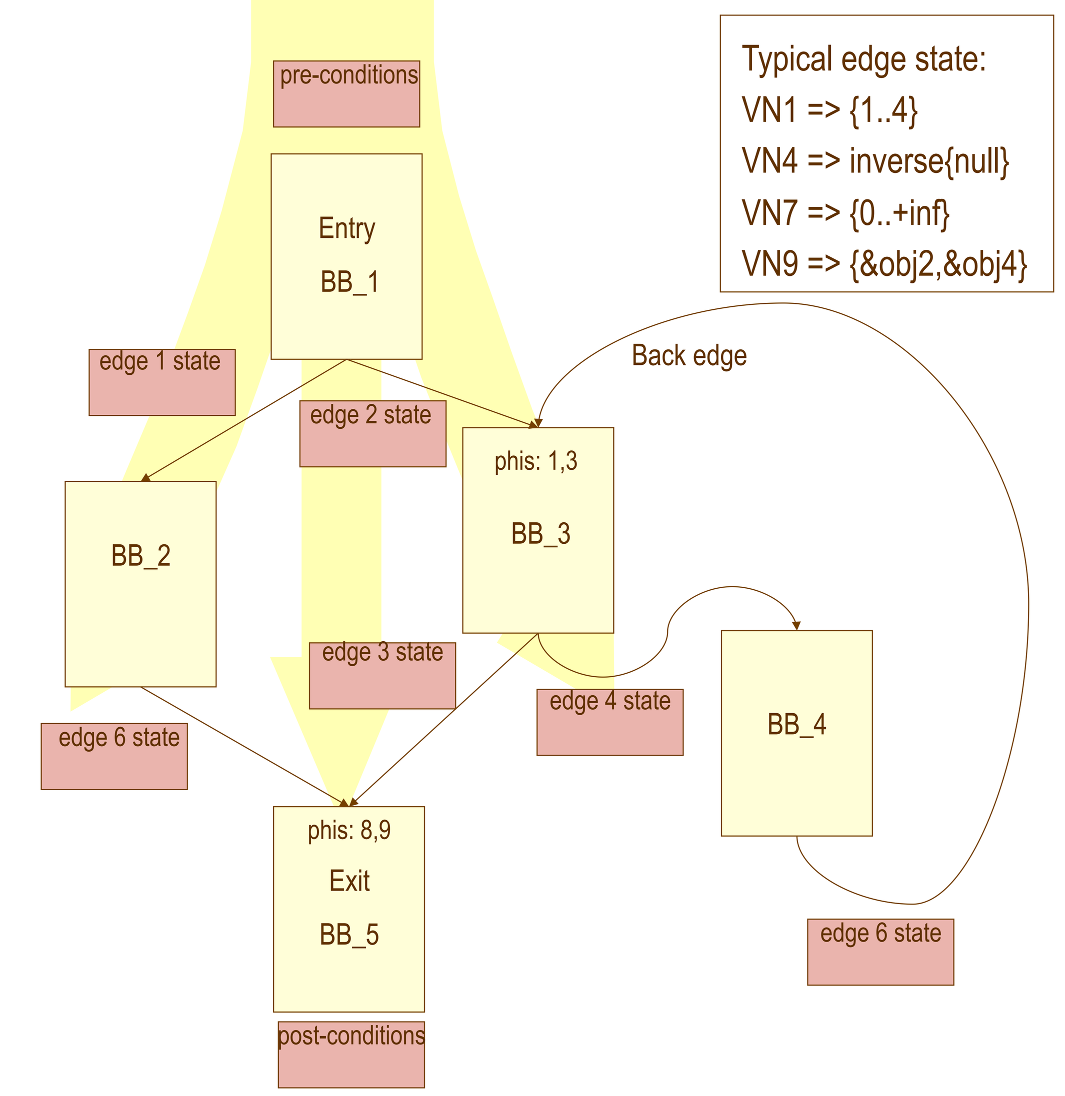
SSA/GVN



- Each procedure converted into Static Single Assignment (SSA) representation, followed by a Global Value Numbering (GVN).
- Single Obj-ID split into multiple Phi Value Numbers (Phi VNs) by Address value number
- "Kappa" value number introduced to represent value of potential alias after assignment, one per address VN
- Flow-sensitive address VNs used to choose between "old" and "new" values for Kappa node

Typical Control Flow Graph during SSA/GVN and PVP

Control-Flow Graph



Possible-Value Set Propagation

PVP

Result of PVP for block:
VN1 => {1..4}
VN4 => inverse{null}
VN7 => {0..+inf}
VN9 => {&obj2,&obj4}

- *Possible Value Set* maintained for each value number, for each *basic block*, which are then constrained by run-time checks
- Constraints *propagated within basic block* until stabilizes, both "bottom up" and "top down," as well as "across" between VNs determined to be related by transitivity or other algebraic equivalence
- Value sets *propagated across basic blocks*, directly, and via "phi" nodes, iterating until fix point reached.
- *Preconditions* determined, so as to *minimize run-time check failures* while attempting to avoid killing off interesting basic blocks
- Iteration performed again to determine *postconditions*
- *"Grand" iteration* performed at procedure level presuming there is mutual recursion (possibly due to indirect/dynamically-dispatched calls)
- Final pass produces *messages* identifying points where *run-time checks might fail*

Influences and Related Work

Influences:

- Building an Optimizing Compiler (R. Morgan)
- Static Single Assignment theory (Cytron et al)
- Ada Compiler Range-Check Elimination

Related work:

- Clousot (CC-Check, Microsoft)
- Infer (Monoidics)

Conclusion

Abstract Interpretation can be re-engineered for *scalability* and *precision* by incorporating the inherently *bottom-up approaches* developed and refined over the years as part of *optimizing compilers*. *Contracts*, in the form of pre/postconditions can be *inferred*, providing a bridge to a *more formal approach* to software development