Whirlwind dataflow analysis engine

Client defines a subclass of `LatticeElmt` (which is a subclass of `AnalysisInfo`) to represent elements of domain

- `<=` (lattice ≤ operator)
- `meet` (lattice g.l.b. operator)

Client defines a subclass of `AnalysisGraph` to specify the graph over which to analyze

- {`Forward`,`Backward`}{`CFG`,`DFG`}`AnalysisGraph` already available

Client defines a subclass of `Analysis` that describes the analysis

- `top_analysis_info` (the top `AnalysisInfo`)
- `analyze(Analysis, AnalysisGraph, IRNode, indexed[LatticeElmt] ):AnalysisAction` (the flow function)
  - Typically many `analyze` multimethods dispatching on different `IRNode` subclasses

Client invokes `analyze_and_transform( Analysis,AnalysisGraph,indexd[AnalysisInfo])` to run the analysis and do all the transformations

Analysis actions

The result of the `analyze` flow function on an `IRNode` is either

- `ContinueAnalysisAction`: propagate a resulting `AnalysisInfo` along successor edge(s)
- `ReplaceAnalysisAction`: replace the `IRNode` with some other sub-`AnalysisGraph`, and restart analysis

`ReplaceAnalysisAction` specifies the transformation to perform as a result of analysis

Also implicitly specifies how to *simulate* the transformation during iterative analysis

- the engine transparently analyzes the replacement graph in lieu of the replaced `IRNode`, to simulate what would happen if the transformation were done

Composed analyses

Whirlwind allows several dataflow analyses to be performed "in parallel"

- interleaved at each `IRNode` operation

If one analysis chooses a transformation, others are reevaluated on the replacement subgraph

- allows improvements of one analysis to improve quality of other analyses, without any explicit accounting in them

Client defines each component analysis as subclass of `ComposableAnalysis`

Client defines a composition of analyses as subclass of {`Forward`,`Backward`}`ComposedAnalysis`

`ComposedAnalysis` is just a regular analysis whose `analyze` flow function invokes each of the component analyses' flow functions in turn

[Lerner, Grove, Chambers, POPL '02]

Features of Whirlwind's dataflow analysis engine

Big idea: separate analyses and transformations, make framework compose them appropriately

- don't have to simulate the effect of transformations during analysis
- can run analyses in parallel if each provides opportunities for the other
  - sometimes can achieve strictly better results this way than if run separately in a loop
- quite drastic transformations supported (e.g. inlining, branch folding) during analysis
- no non-local transformations (e.g. code motion) supported

Makes no sacrifices of precision for speed

- has few speed-related optimizations

Cobalt

The next generation: specify dataflow analyses in a specialized declarative language

+ allow mechanical proof of correctness of optimizations!
+ allow mechanical integration, compilation down to efficient code!

Have a simple version of Cobalt running in Whirlwind

- Cobalt-2 being designed now

[Lerner, Millstein, Chambers, PLDI '03 (Best Paper Award)]