Detecting deadlocks using static analysis in .NET

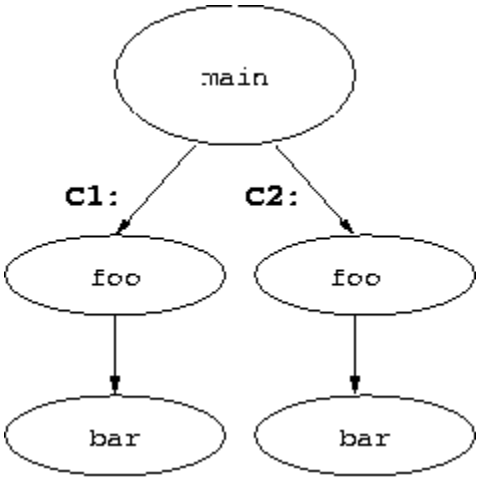
Filip Navara

filip.navara@gmail.com

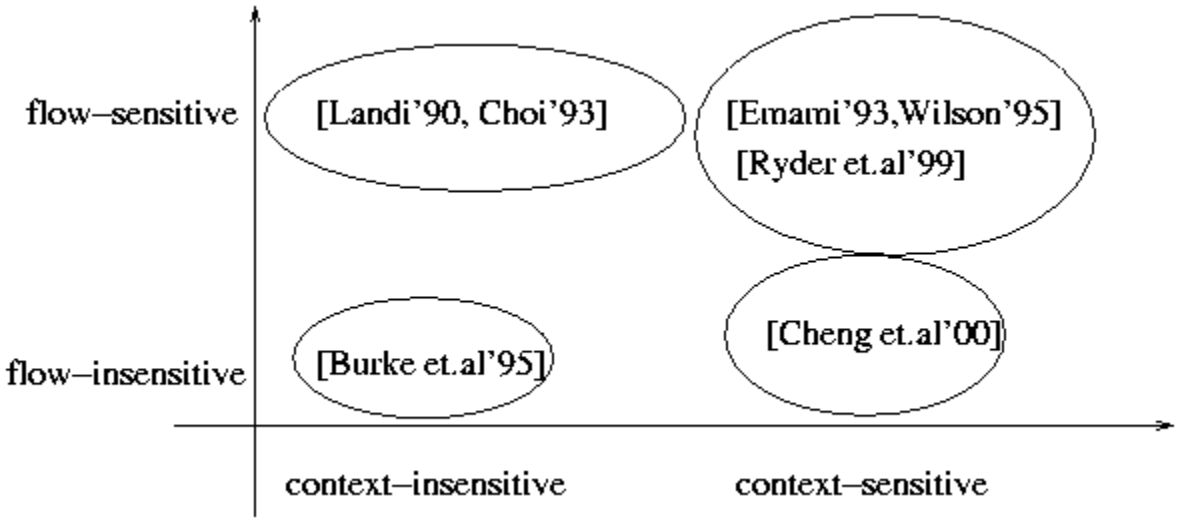
What did I do last week?

- Started writing down a paper about the static code analysis concepts
  - Still fighting LaTeX...
  - Very incomplete
  - Work in progress

Invocation graph



Interprocedural data-flow algorithms



Interprocedural data-flow algorithms

	Landi'90	Emami'93	Wilson'95	Burke'95	Choi'93	Ryder'99	Cheng'00
Alias representation	Complete pairs	Point-to	Point-to	Compact pairs	Compact pairs	Point-to	Point-to
Framework	ICFG	IG	IG	PCG	PCG	PCG	PCG
Target language and features	Some C features	Some C features	All C features, use lower level location set, <v,offset,stide>, simple pointer arithmetic	Some C features	Some C features	Some C and C++ features	All C features, use lower level offset to access fields
Benchmarks tested	Non-SPEC benchmarks up to 4663 lines source	Non-SPEC benchmarks up to 2779 lines source	SPEC92 +others up to 24,000 lines source	SPEC92 + others up to 29,000 lines source	Same as Burke'95	Non-SPEC benchmarks up to 6000 lines source	SPEC92 and 95 benchmarks up to 200,000 lines source (like gcc)
modular	No	No	No	No	No	Yes, SCC	Yes, SCC

What do I plan to do next week(s)?

- Get the GraphViz LaTeX extension working...
- Finish the intra-procedural part of the paper and start filling up the holes in the inter-procedural chapter...