[Dataflow Analysis of Array and Scalar References](https://link-springer-com.proxy.library.uu.nl/article/10.1007/BF01407931)

Given an array cell, which of several statements is the source of the value contained therein at a given instant in the execution of a program.

[Data Flow Analysis In Software Reliability](https://dl-acm-org.proxy.library.uu.nl/doi/pdf/10.1145/356674.356676)

Basic terminology from graph theory and from data flow analysis. Has some references to the initial establishment of data flow analysis. Could be useful, but old (1976).

[Analyzing Aliases of Reference Formal Parameters](https://dl-acm-org.proxy.library.uu.nl/doi/pdf/10.1145/318593.318658)

Techniques for aliasing. Is this a potentially viable solution to reference equality?

[A generic framework for multilanguage analysis](http://dspace.unive.it/bitstream/handle/10579/23717/956516-1255497.pdf?sequence=2)

A method of applying abstract analysis on multiple different languages, could be interesting later on.

[A Program Data Flow Analysis Procedure](https://dl-acm-org.proxy.library.uu.nl/doi/pdf/10.1145/360018.360025)

One of the original papers on data flow analysis

[Dimensions of Precision in Reference Analysis of Object-Oriented Programming Languages](https://link-springer-com.proxy.library.uu.nl/chapter/10.1007/3-540-36579-6_10)

Lays out a lot of different reference analysis types, seems like a very good starting point.

[Language design and analyzability: A retrospective](https://onlinelibrary-wiley-com.proxy.library.uu.nl/doi/full/10.1002/spe.1133?casa_token=WKI19BFlofgAAAAA%3AaPyz0XkPj1M5CEzBL3k5Y-xDc32G56E75Xfal3EC7Z91ANFCMqNM7ihSdtrMxFs1FIMwFJRsgZaFP6c)

Lays out inherent issues with Dataflow analysis, section on pointers.

[Path-sensitive value-flow analysis](https://dl-acm-org.proxy.library.uu.nl/doi/10.1145/268946.268966)

Looks specifically at recomputation of a value with multiple, synonymous names.

[Extent analysis of data fields](https://link-springer-com.proxy.library.uu.nl/chapter/10.1007/3-540-58485-4_42)

Might be relevant if concurrency is important.

[Scalar replacement in the presence of conditional control flow](https://onlinelibrary-wiley-com.proxy.library.uu.nl/doi/10.1002/spe.4380240104)

Arrays are often treated as scalars in data flow analysis, failing to locate the array elements to registers.

[Ah Ahead-of-time Yet Context-Sensitive Points-to Analysis for Java](https://www-sciencedirect-com.proxy.library.uu.nl/science/article/pii/S1571066109004630?via%3Dihub)

Unique approach to points-to analysis by doing it ahead of time. Also specifically Java. Hard to understand intro, but maybe worth looking at.

[Using XBDDs and ZBDDs in points-to analysis](https://onlinelibrary-wiley-com.proxy.library.uu.nl/doi/10.1002/spe.895)

Space-efficient method of representing relations in point-to or reference analyses. Relations is what this is all about, right?

[Fast and Precise Points-to Analysis](https://ieeexplore-ieee-org.proxy.library.uu.nl/document/4637546)

Fast and memory efficient points-to analysis method.

[Contribution-based call stack abstraction for call string based pointer analysis](https://www-sciencedirect-com.proxy.library.uu.nl/science/article/pii/S0950584910002053?via%3Dihub)

Analysis for function pointers based on call string. Improves on earlier call string based techniques by taking different contributions into account.

[Source Code Analysis: A Road Map](https://ieeexplore-ieee-org.proxy.library.uu.nl/stamp/stamp.jsp?tp=&arnumber=4221615&tag=1)

General overview of source code analysis, as well as speculation about the future.

[Towards Path-Sensitive Points-to Analysis](https://ieeexplore-ieee-org.proxy.library.uu.nl/document/4362898)

WPP2G and P2SSA are very useful for points-to analysis, this paper improves on those techniques.

[Pick your contexts well: understanding object-sensitivity](https://dl-acm-org.proxy.library.uu.nl/doi/10.1145/1926385.1926390)

Points out sub-optimal choice of contexts, so perhaps this could be used as a view of what not to do. Also looks at “type-sensitivity”, could be an interesting avenue to explore. Has a cool video explaining the paper (I think).

[Refinement-based context-sensitive points-to analysis for Java](https://dl-acm-org.proxy.library.uu.nl/doi/10.1145/1133981.1134027)

Points-to analysis with relation specifically to downcasting. Who knows?

[P/Taint: unified points-to and taint analysis](https://dl-acm-org.proxy.library.uu.nl/doi/10.1145/3133926)

Taint analysis and points-to analysis at once. What is taint analysis?

[Pointer-induced Aliasing: A Problem Classification](https://dl-acm-org.proxy.library.uu.nl/doi/pdf/10.1145/99583.99599)

Has references to calculation of aliases for FORTRAN, but when pointers are added, the problem of computing aliases becomes NP-hard. Shows which aspects of alias problems are hard and need to be approximated. Because not all languages have pointers, this could be used for the basis of a split when discussing viability or analysis techniques, between those two types of languages.

[Interprocedural modification side effect analysis with pointer aliasing](https://dl-acm-org.proxy.library.uu.nl/doi/abs/10.1145/173262.155096)

Has a reference to an approximation algorithm for pointer aliases.

Sources from Kotlin specification:

[Control flow analysis](https://dl-acm-org.proxy.library.uu.nl/doi/abs/10.1145/390013.808479)

[Principles of program analysis](https://books.google.nl/books?hl=en&lr=&id=YseqCAAAQBAJ&oi=fnd&pg=PA1&dq=Principles+of+program+analysis&ots=yC_IJxj8xH&sig=BJuUJEEkjeLm0X9CUeuzpoCo2UI&redir_esc=y#v=onepage&q=Principles%20of%20program%20analysis&f=false)

[Monotone data flow analysis frameworks](https://link-springer-com.proxy.library.uu.nl/article/10.1007/BF00290339)

Important search terms:

Whole Program Points-to Graph (WPP2G)

Points-to SSA (P2SSA)

Questions:

Is parallelism something I have to look at for this paper, or is it beyond scope?

Kotlin has stable references, that will not change,