

CLAIMS

What is claimed is:

1. A dependency analyzer for use with a data processing system comprising:
 - at least one computer hardware processor;
 - at least one non-transitory computer-readable storage medium storing processor executable instructions that, when executed by the at least one computer hardware processor, cause the at least one computer hardware processor to perform:
 - for each of a plurality of programs configured for execution by the data processing system, constructing a first data structure based on parsing at least a portion of the program, wherein the first data structure reflects operations in the program, and wherein the plurality of programs are written in a plurality of programming languages;
 - for each of a plurality of the first data structures:
 - selecting operations reflected in the first data structure that impact a dependency on one or more data elements, such that operations that do not impact a dependency are not selected, wherein selecting the operations comprises identifying from a program, at least a portion of which had been parsed to construct the first data structure, indications of operations that impact a control flow or a data flow dependency using a tool selected based on a programming language in which the program was written; and
 - for each selected operation, recording in a second data structure an indication of the selected operation by recording dependency constructs selected from a set of dependency constructs, wherein the set of dependency constructs, for each of the plurality of

programs, is not dependent on a language in which the program is written;

processing a plurality of second data structures in which the indications of selected operations are recorded, wherein the processing comprises identifying control flow or data flow dependencies on data elements, said dependencies being those that would arise during execution of any of the plurality of programs; and

recording the identified control flow or data flow dependencies in a dependency data structure, wherein the dependency data structure comprises data indicating data elements in the plurality of programs structured so as to indicate control flow or data flow dependencies between the data elements.

2. The dependency analyzer of claim 1, wherein:
the dependencies comprise control flow dependencies and data flow dependencies.
3. The dependency analyzer of claim 1, wherein:
the dependencies comprise data flow dependencies.
4. The dependency analyzer of claim 1, wherein:
the data processing system comprises a plurality of language dependent parsers, and
constructing the first data structure comprises selecting, for each of the plurality of programs, a language dependent parser based on a language in which the program is written.
5. The dependency analyzer of claim 4, wherein:
the plurality of programs are written in a plurality of source languages.

6. The dependency analyzer of claim 5, wherein:
the source languages include SQL.
7. The dependency analyzer of claim 5, wherein:
the source languages include a graphical programming language expressing a data flow graph.
8. The dependency analyzer of claim 5, wherein:
the source languages include SQL and a graphical programming language expressing a data flow graph.
9. The dependency analyzer of claim 4, wherein:
the data processing system comprises a plurality of language dependent tools for extracting from the first data structures indications of operations that impact a control flow or a data flow dependency.
10. The dependency analyzer of claim 1, wherein:
for each of the plurality of constructed first data structures, recording in the second data structure indications of selected operations comprises recording a construct from a set of constructs.
11. The dependency analyzer of claim 10, wherein:
the set of constructs comprises two or more of the following: constructs representing assignment, procedure definition, procedure return, a procedure call site, conditional control flow in an “if...else” format, and conditional control flow in a “while ...do” format.
12. The dependency analyzer of claim 10, wherein:

the set of constructs consists essentially of constructs representing assignment, procedure definition, procedure return, a procedure call site, conditional control flow in an “if...else” format, and conditional control flow in a “while ...do” format.

13. The dependency analyzer of claim 2, further comprising:
at least one tool that accesses the dependency data structure and presents, through a user interface, information about a data entity or code block that is dependent on a data element.
14. The dependency analyzer of claim 2, wherein:
the data processing system further comprises a plurality of data sources;
the plurality of programs are configured to perform transformations on data in the plurality of data sources; and
the plurality of data sources are heterogeneous.
15. The dependency analyzer of claim 14, wherein:
the plurality of data sources comprise data sources that are at least one of an ORACLE database or an SAP database or a dataset stored in the HADOOP Distributed Files System.
16. A method of operating a dependency analyzer configured for use with a data processing system, the method comprising:
using at least one computer hardware processor to perform, for a plurality of programs configured for execution within the data processing system and specifying data processing operations in programming languages, wherein the plurality of programs are written in a plurality of programming languages:
constructing a respective first data structure for the plurality of programs by parsing at least a portion of each program;

for each of a plurality of the first data structures:

selecting operations reflected in the first data structure that impact a dependency on one or more data elements, wherein selecting the operations comprises identifying from a program, at least a portion of which had been parsed to construct the first data structure, indications of operations that impact a control flow or a data flow dependency using a tool selected based on a programming language in which the program was written; and

for each selected operation, recording in a second data structure an indication of the selected operation, wherein the indications are recorded in the second data structure using a set of dependency constructs that is common to all of the second data structures;

processing the second data structures in which the indications of selected operations are recorded, wherein the processing comprises identifying control flow or data flow dependencies on data elements that would arise during execution of any of the plurality of programs; and

recording in a dependency data structure the identified control flow or data flow dependencies, wherein the dependency data structure comprises data indicating data elements in the plurality of programs structured so as to indicate control flow or data flow dependencies between the data elements.

17. The method of claim 16, wherein:
the dependencies are data flow dependencies.

18. The method of claim 16, wherein:
the dependencies comprise control flow dependencies data flow dependencies.

19. The method of claim 16, wherein:
constructing a respective first data structure for the plurality of programs comprises selecting parsers for the plurality of programs based on languages in which the plurality of programs are written.
20. The method of claim 16, wherein:
the plurality of programs are written in a plurality of source languages, including SQL and a graphical programming language expressing a data flow graph; and
selecting parsers comprises selecting a parser for each of the plurality of source languages.
21. The method of claim 20, wherein:
recording in the second data structures comprises constructing second data structures from the set of dependency constructs for programs written in each of the plurality of source languages.
22. The method of claim 16, wherein:
the first data structures are language dependent; and
the second data structures are language independent.
23. The method of claim 16, wherein:
recording in the second data structure indications of the selected operations comprises extracting from the first data structure indications of operations that impact a control flow or a data flow dependency such that information about operations in the first data structure that do not impact a control flow or a data flow dependency are omitted from the second data structure.
24. The method of claim 16, wherein:

the data processing system comprises a plurality of language dependent tools for extracting from the first data structures indications of operations that impact a control flow or a data flow dependency.

25. The method of claim 16, wherein:

the set of dependency constructs comprises two or more of the following: constructs representing assignment, procedure definition, procedure return, a procedure call site, conditional control flow in an “if...else” format, and conditional control flow in a “while ...do” format.

26. A method of operating a dependency analyzer for use in conjunction with a data processing system, the method comprising:

using at least one computer hardware processor to perform, for a plurality of programs specifying data processing operations within the data processing system in programming languages, wherein the plurality of programs use a plurality of programming languages:

constructing respective first data structures for the plurality of programs by parsing at least a portion of each program;

for each of a plurality of the first data structures:

selecting operations reflected in the first data structure that impact a control flow dependency on one or more data elements, wherein selecting the operations comprises identifying from a program, at least a portion of which had been parsed to construct the first data structure, indications of operations that impact a control flow dependency using a tool selected based on a programming language in which the program was written; and

for each selected operation, recording in a second data structure an indication of the selected operation, wherein the indications are recorded in the second data structure using a set of dependency constructs that is common to all of the second data structures;

processing the second data structures, wherein the processing comprises identifying control flow dependencies on the one or more data elements as a result of processing in any of the plurality of programs; and

recording in a dependency data structure the identified control flow dependency on the one or more data elements.

27. The method of claim 26, wherein:

the set of dependency constructs comprises conditional control flow in an “if...else” format.

28. The method of claim 26, wherein:

the set of dependency constructs comprises conditional control flow in a “while ...do” format.

29. The method of claim 26, further comprising:

for each of the plurality of the constructed first data structures, selectively recording in the second data structure indications of operations in the first data structure that create a data flow dependency, wherein the indications are recorded in the second data structure using a set of dependency constructs that is common to all of the second data structures.

30. At least one non-transitory computer-readable storage medium storing processor executable instructions that, when executed by at least one computer hardware processor, cause the at least one computer hardware processor to perform a method, for a plurality of programs specifying data processing operations in respective programming languages, wherein the plurality of programs use a plurality of programming languages, the method comprising:

constructing a respective first data structure for the plurality of programs by parsing at least a portion of each program;

for each of a plurality of the constructed first data structures:

extracting from the first data structure operations reflected in the first data structure that impact a control flow or a data flow dependency on one or more data elements using a tool selected based on a programming language in which a program, at least a portion of which had been parsed to construct the first data structure, was written, and

recording indications of the extracted operations in a second data structure, wherein the indications of the extracted operations are recorded in the second data structure using a set of dependency constructs that is common to all of the second data structures;

processing the second data structures, wherein the processing comprises identifying dependencies on data elements as a result of processing in any of the plurality of programs; and

recording in a dependency data structure the identified dependencies.

31. The non-transitory computer-readable storage medium of claim 30, wherein:
constructing a respective first data structure for the plurality of programs comprises selecting parsers for the plurality of programs based on languages in which the plurality of programs are written.

32. The non-transitory computer-readable storage medium of claim 31, wherein:
the plurality of programs are written in a plurality of source languages, including SQL and a graphical programming language expressing a data flow graph;
and
selecting parsers comprises selecting a parser for each of the plurality of source languages.

33. The non-transitory computer-readable storage medium of claim 32, wherein:

extracting indications of operations in each of the plurality of the constructed first data structures comprises constructing second data structures from the set of dependency constructs for programs written in each of the plurality of source languages.

34. The non-transitory computer-readable storage medium of claim 30, wherein:
the first data structures are language dependent; and
the second data structures are language independent.

35. The non-transitory computer-readable storage medium of claim 30, wherein:
extracting indications of operations in the first data structure comprises selectively recording in the second data structure indications of operations in the first data structure that impact a dependency such that information about operations in the first data structure that do not impact a dependency are omitted from the second data structure.

36. The non-transitory computer-readable storage medium of claim 30, wherein:
the data processing system comprises a plurality of language dependent tools for lowering the first data structures, and
extracting indications of operations in the first data structure comprises, for each of the plurality first data structures, selecting a language dependent tool for extracting indications of operations in the first data structure based on a language in which a program that was parsed to construct the first data structure is written.

37. The non-transitory computer-readable storage medium of claim 30, wherein:
the set of dependency constructs comprises two or more of the following:
constructs representing assignment, procedure definition, procedure return, a procedure call site, conditional control flow in an “if...else” format, and conditional control flow in a “while ...do” format.

38. A dependency analyzer for use with a data processing system, the dependency analyzer comprising:

means for constructing a first data structure representing each of a plurality of programs, wherein the first data structures reflect operations in respective programs of the plurality of programs, and wherein the plurality of programs are written in a plurality of programming languages;

means for identifying from each first data structure indications of operations that impact a control flow or a data flow dependency using a tool selected based on a programming language in which a program that had been parsed to construct the first data structure was written;

means for recording in a second data structure indications of operations in the first data structure that impact a control flow or a data flow dependency on one or more data elements such that information about operations in each of the plurality of the constructed first data structures that do not impact a dependency are omitted from the second data structure;

means for processing the second data structures to identify control flow or a data flow dependencies on the one or more data elements as a result of execution of operations in any of the plurality of programs; and

means for recording in a dependency data structure the identified control flow or data flow dependencies on the one or more data elements.