CLAIMS

What is claimed is:

1. ~~A~~ At least one non-transitory computer readable medium comprising instructions that, when executed by at least one processor, perform a method of converting a program written in a first programming language into a dataflow graph, wherein the program comprises data operations and a control flow indicating a sequence of execution of the data operations, the method comprising:

analyzing the program using the at least one processor and using a grammar stored by the at least one computer readable medium, wherein the grammar indicates:

how to identify data operations within programs written in the first programming language, how to identify dataflow graph components corresponding to data operations within programs written in the first programming language, and how to configure dataflow graph components that correspond to data operations within programs written in the first programming language, said analyzing comprising:

identifying, based on the grammar indicating how to identify data operations within the program, a plurality of sequential steps of the program, wherein each step of the plurality of steps comprises one or more data operations of the program;

generating a catalog storing control flow relationships between steps of the plurality of identified steps of the program;

for each of the identified plurality of steps of the program, based on data operations of the step and based on the grammar~~,~~ identifying one of a plurality of dataflow graph component types as corresponding to the step, ~~thereby~~ identifying a plurality of dataflow graph components each corresponding to a respective step of the plurality of steps of the program; and

configuring the plurality of dataflow graph components based on respective corresponding steps of the plurality of steps of the program and based on the grammar indicating how to configure the plurality of dataflow graph components, thereby producing a plurality of configured dataflow graph components; wherein configuring the plurality of dataflow graph components comprises generating configuration data for at least one of the plurality of dataflow graph components and associating the configuration data with the at least one dataflow graph component; and

generating, using the at least one processor, a dataflow graph comprising:

the plurality of configured dataflow graph components; and

one or more flows representing flows of data records between the one or more dataflow graph components,

wherein the one or more dataflow graph components are connected via the one or more flows in an order according to the generated catalog.

2. The non-transitory computer readable medium of claim 1, wherein the program is a Statistical Analysis System (SAS) program.

3. The non-transitory computer readable medium of claim 1, wherein analyzing the program further comprises identifying a macro defined in the program, and wherein configuring the plurality of dataflow graph components further comprises generating at least one parameter set based on the macro definition.

4. The non-transitory computer readable medium of claim 1, wherein the program comprises comments and wherein configuring the plurality of dataflow graph components comprises configured at least some of the plurality of dataflow graph component to include the comments.

5. The non-transitory computer readable medium of claim 1, wherein the generated dataflow graph comprises first and second groups of dataflow graph components configured to execute in parallel with one another.

6. The non-transitory computer readable medium of claim 5, wherein:

the program comprises a first group of one or more of the plurality of data operations and a second group of one or more of the plurality of data operations,

the first group of data operations and the second group of data operations have no dependency on one another,

the first group of dataflow graph components performs the first group of data operations, and

the second group of dataflow graph components performs the second group of data operations.

7. The non-transitory computer readable medium of claim 1, wherein the program comprises whitespace between symbols of the program and wherein configuring the plurality of dataflow graph components preserves whitespace between portions of generated instructions corresponding to the symbols.

8. The non-transitory computer readable medium of claim 1, wherein analyzing the program further comprises identifying a subroutine defined in the program, and wherein the method further comprises generating at least one subgraph based on the subroutine definition.

9. The non-transitory computer readable medium of claim 8, wherein the subroutine is defined with one or more input values whose values alter the executable behavior of the subroutine, and wherein the generated at least one subgraph is generated with at least one subgraph parameter corresponding to the one or more input values of the subroutine.

10. The non-transitory computer readable medium of claim 1, wherein each step of the plurality of steps comprises a plurality of data operations of the program.

11. A computer-implemented method of automatically converting a program written in a first programming language into a dataflow graph, wherein the program comprises data operations and a control flow indicating a sequence of execution of the data operations, the method comprising:

analyzing the program using the at least one processor and using a grammar stored by ~~the~~ at least one computer readable medium, wherein the grammar indicates:

how to identify data operations within programs written in the first programming language, how to identify dataflow graph components corresponding to data operations within programs written in the first programming language, and

how to configure dataflow graph components that correspond to data operations within programs written in the first programming language, said analyzing comprising:

identifying, based on the grammar indicating how to identify data operations within the program, a plurality of sequential steps of the program, wherein each step of the plurality of steps comprises one or more data operations of the program;

generating a catalog storing control flow relationships between steps of the plurality of identified steps of the program;

for each of the identified plurality of steps of the program, based on data operations of the step and based on the grammar~~,~~ identifying one of a plurality of dataflow graph component types as corresponding to the step, ~~thereby~~ identifying a plurality of dataflow graph components each corresponding to a respective step of the plurality of steps of the program; and

configuring the plurality of dataflow graph components based on respective corresponding steps of the plurality of steps of the program and based on the grammar indicating how to configure the plurality of dataflow graph components, thereby producing a plurality of configured dataflow graph components , wherein configuring the plurality of dataflow graph components comprises generating configuration data for at least one of the plurality of dataflow graph components and associating the configuration data with the at least one dataflow graph component; and

generating, using the at least one processor, a dataflow graph comprising: the plurality of configured dataflow graph components; and

one or more flows representing flows of data records between the one or more dataflow graph components,

wherein the one or more dataflow graph components are connected via the one or more flows in an order according to the generated catalog.

12. The method of claim 11, wherein the program is a Statistical Analysis System (SAS) program.

13. The method of claim 11, wherein analyzing the program further comprises identifying a macro defined in the program, and wherein configuring the plurality of dataflow graph components further comprises generating at least one parameter set based on the macro definition.

14. The method of claim 11, wherein the program comprises comments and wherein configuring the plurality of dataflow graph components comprises configured at least some of the plurality of dataflow graph component to include the comments.

15. The method of claim 11, wherein the generated dataflow graph comprises first and second groups of dataflow graph components configured to execute in parallel with one another.

16. The method of claim 15, wherein:

the program comprises a first group of one or more of the plurality of data operations and a second group of one or more of the plurality of data operations,

the first group of data operations and the second group of data operations have no dependency on one another,

the first group of dataflow graph components performs the first group of data operations, and

the second group of dataflow graph components performs the second group of data operations.

17. The method of claim 11, wherein the program comprises whitespace between symbols of the program and wherein configuring the plurality of dataflow graph components preserves whitespace between portions of generated instructions corresponding to the symbols.

18. The method of claim 11, wherein analyzing the program further comprises identifying a subroutine defined in the program, and wherein the method further comprises generating at least one subgraph based on the subroutine definition.

19. The method of claim 18, wherein the subroutine is defined with one or more input values whose values alter the executable behavior of the subroutine, and wherein the generated at least one subgraph is generated with at least one subgraph parameter corresponding to the one or more input values of the subroutine.

20. The method of claim 11, wherein each step of the plurality of steps comprises a plurality of data operations of the program.

21. A system for automatically converting a program written in a first programming language into a dataflow graph, wherein the program comprises data operations and a control flow indicating a sequence of execution of the data operations,

the system comprising: means for analyzing the program using the at least one processor and using a grammar stored by ~~the~~ at least one computer readable medium, wherein the grammar indicates:

how to identify data operations within programs written in the first programming language, how to identify dataflow graph components corresponding to data operations within programs written in the first programming language, and how to configure dataflow graph components that correspond to data operations within programs written in the first programming language, said analyzing comprising:

identifying, based on the grammar indicating how to identify data operations within the program, a plurality of sequential steps of the program, wherein each step of the plurality of steps comprises one or more data operations of the program;

generating a catalog storing control flow relationships between steps of the plurality of identified steps of the program;

for each of the identified plurality of steps of the program, based on data operations of the step and based on the grammar, identifying one of a plurality of dataflow graph component types as corresponding to the step, ~~thereby~~ identifying a plurality of dataflow graph components each corresponding to a respective step of the plurality of steps of the program; and

configuring the plurality of dataflow graph components based on respective corresponding steps of the plurality of steps of the program and based on the grammar indicating how to configure the plurality of dataflow graph components, thereby producing a plurality of configured dataflow graph components, wherein configuring the plurality of dataflow graph components comprises generating configuration data for at least one of the plurality of dataflow graph components and associating the configuration data with the at least one dataflow graph component; and

means for generating a dataflow graph, the dataflow graph comprising:

the plurality of configured dataflow graph components; and

one or more flows representing flows of data records between the one or more dataflow graph components,

wherein the one or more dataflow graph components are connected via the one or more flows in an order according to the generated catalog.