CLAIMS

1. A compiler method for a smart contract, the method comprising:

by the smart contract deployed on a blockchain:

receiving an input string having at least one agreement condition and at least one action;

accepting the input string;

parsing, using a parser, the input string into an object structure, wherein the object structure is in a string format;

determining a byte sequence program for the object structure;

mapping the byte sequence program into at least one executable smart contract representation of the object structure executable by the blockchain, wherein the at least one executable smart contract representation implements at least one function of a programming language, wherein in response to receiving a request for executing the at least one function of the programming language, the at least one executable smart contract representation for the at least one function is executed and the at least one action is performed;

receiving a second input string for updating a segment of at least one of the agreement conditions;

parsing, using the parser, the second input string into a second object structure comprising the segment;

determining a second byte sequence program for the second object structure; and

updating the byte sequence program with the second byte sequence program.

2. The compiler method of claim 1, wherein the segment is a segment of a function.

3. The compiler method of claim 1, wherein the at least one executable smart contract representation implements native functions or native operations.

4. The compiler method of claim 3, wherein the native functions or the native operations include:

one or more binary comparators including at least one of equal, not equal, greater than or less than;

one or more set operations including at least one of logical AND, logical OR, COUNT, or SUM;

one or more compositional operators including at least one of plus, minus, multiplication, or division; or

one or more loops including while-loops, for-loops, counters, if-statements, switch statements.

5. The compiler method of claim 3, wherein the at least one executable smart contract representation implements a data source function including at least one of logical TRUE, logical FALSE, or a read-only variable.

6. The compiler method of claim 1, wherein the object structure is a nested tree, each leaf of the nested tree being another object structure.

7. The compiler method of claim 1, wherein the input string is in a recursive data structure.

8. The compiler method of claim 7, wherein the accepting of the input string is based on a computing device or a user providing the input string having permission to provide the input string.

9. The compiler method of claim 7, wherein the recursive data structure is a domain-specific language.

10. The compiler method of claim 1, wherein the byte sequence program is an ordered sequence of bytecodes for executing the at least one agreement condition and the at least one action.

11. The compiler method of claim 1, further comprising, by the smart contract deployed on the blockchain, evaluating the at least one executable smart contract representation in response to the at least one agreement condition being satisfied.

12. The compiler method of claim 1, wherein the at least one executable smart contract representation is executable by an Ethereum Virtual Machine (EVM) of the blockchain.

13. The compiler method of claim 1, wherein the receiving the second input string, the parsing the second input string, and the determining the second byte sequence program is performed by a master smart contract which is the smart contract deployed on the blockchain.

14. The compiler method of claim 1, wherein the updating of the byte sequence program is updated exclusive of any other segments and any other agreement conditions.

15. The compiler method of claim 1, wherein the second string input is for further updating an amount of at least one of the actions.

16. The compiler method of claim 1, wherein the second string input is for further updating one or more further segments of at least one of the agreement conditions.

17. The compiler method of claim 1, wherein the input string, the object structure, and the byte sequence program represent a workflow is an ordered sequence.

18. The compiler method of claim 1, further comprising:

sending, from a computing device to the smart contract on the blockchain, the second input string;

receiving, from the smart contract on the blockchain by the computing device or another computing device, confirmation of execution of the at least one smart contract representation.

19. The compiler method of claim 1, wherein the receiving the second input string is performed after the mapping.

20. A computer-readable medium having tangibly stored thereon computer-executable instructions that, in response to execution by a processor of a compiler system, cause the compiler system to perform the compiler method of any one of claims 1-19.