ALGORITHMS AND PROCEDURES FOR BUILDING AND RUNNING SYSTEMS OF COMPLEX ANALYTICS", docket no. LIY01.004.

[0004] Applicant hereby claims priority under 35 U.S.C. § 119 to United States provisional patent application no. 62/894,001, filed August 30, 2019, entitled "DECLARATIVE SYNTAX AND RELATED METHODS, INTERLACE, ALGORITHMS AND PROCEDURES FOR BUILDING AND RUNNING SYSTEMS OF COMPLEX ANALYTICS", docket no. 3200-101PV3.

[0005] The entire contents of the aforementioned applications are herein expressly incorporated by reference in their entirety.

FIELD

[0006] The present disclosure is directed generally to compilers. Compilers are an essential computer technology that allows computer-implemented generation of processor-executable instructions from (e.g., human-readable) source code. The DSGRCE implements improvements to the art.

BACKGROUND

[0007] A compiler translates computer code written in one programming language into a different language.

SUMMARY OF INVENTION

[0007A] According to an aspect of the present disclosure, there is provided a distributed system generating rule compiler apparatus, comprising: a memory; a

processor in communication with the memory, and configured to issue a plurality of processing instructions stored in the memory, wherein the processor issues instructions to: (a) obtain, via the processor, a distributed system execution request data structure; (b) determine, via the processor, an output rule and the output rule's set of rule parameters specified via the distributed system execution request data structure for a requested calculation, wherein a current rule is initialized to the output rule; (c) query, via the processor, a rule lookup table data structure to determine a set of matching rules, corresponding to the current rule, based on the current rule's identification tuple; (d) select, via the processor, the best matching rule from the set of matching rules, wherein the best matching rule's set of rule parameters best corresponds to the current rule's set of rule parameters; (e) generate, via the processor, a logical dependency graph data structure by adding logical dependency graph nodes and logical dependency graph edges corresponding to the best matching rule, precedent rules of the best matching rule, and precedent rules of each precedent rule; (f) determine, via the processor, an execution complexity gauge value associated with the generated logical dependency graph data structure; (g) determine, via the processor, a set of distributed worker processes to be utilized to execute the requested calculation; (h) divide, via the processor, the generated logical dependency graph data structure into a set of subgraphs based on the execution complexity gauge value, wherein each worker process in the set of distributed worker processes is assigned a subgraph from the set of subgraphs; (i) initialize, via the processor, each respective worker process in the set of distributed worker processes

with the subgraph assigned to the respective worker process; (j) coordinate, via the processor, execution of the requested calculation by worker processes in the set of distributed worker processes, wherein each respective worker process calculates results for logical dependency graph nodes in the subgraph assigned to the respective worker process; and (k) obtain, via the processor, a computation result of the logical dependency graph node corresponding to the output rule, from the worker process in the set of distributed worker processes that is assigned the subgraph that contains the logical dependency graph node corresponding to the output rule.

[0007B] According to another aspect of the present disclosure, there is provided a distributed system generating rule compiler processor-readable non-transitory physical medium storing processor-issuable instructions to: (a) obtain, via the processor, a distributed system execution request data structure; (b) determine, via the processor, an output rule and the output rule's set of rule parameters specified via the distributed system execution request data structure for a requested calculation, wherein a current rule is initialized to the output rule; (c) query, via the processor, a rule lookup table data structure to determine a set of matching rules, corresponding to the current rule, based on the current rule's identification tuple; (d) select, via the processor, the best matching rule from the set of matching rules, wherein the best matching rule's set of rule parameters best corresponds to the current rule's set of rule parameters; (e) generate, via the processor, a logical dependency graph data structure by adding logical dependency graph nodes and logical dependency graph edges corresponding to the best matching rule, precedent

rules of the best matching rule, and precedent rules of each precedent rule; (f) determine, via the processor, an execution complexity gauge value associated with the generated logical dependency graph data structure; (g) determine, via the processor, a set of distributed worker processes to be utilized to execute the requested calculation; (h) divide, via the processor, the generated logical dependency graph data structure into a set of subgraphs based on the execution complexity gauge value, wherein each worker process in the set of distributed worker processes is assigned a subgraph from the set of subgraphs; (i) initialize, via the processor, each respective worker process in the set of distributed worker processes with the subgraph assigned to the respective worker process; (j) coordinate, via the processor, execution of the requested calculation by worker processes in the set of distributed worker processes, wherein each respective worker process calculates results for logical dependency graph nodes in the subgraph assigned to the respective worker process; and (k) obtain, via the processor, a computation result of the logical dependency graph node corresponding to the output rule, from the worker process in the set of distributed worker processes that is assigned the subgraph that contains the logical dependency graph node corresponding to the output rule.

[0007C] According to yet another aspect of the present disclosure, there is provided a processor-implemented distributed system generating rule compiler method, comprising executing processor-implemented instructions to: (a) obtain, via the processor, a distributed system execution request data structure; (b) determine, via the processor, an output rule and the output rule's set of rule parameters specified

via the distributed system execution request data structure for a requested calculation, wherein a current rule is initialized to the output rule; (c) query, via the processor, a rule lookup table data structure to determine a set of matching rules, corresponding to the current rule, based on the current rule's identification tuple; (d) select, via the processor, the best matching rule from the set of matching rules, wherein the best matching rule's set of rule parameters best corresponds to the current rule's set of rule parameters; (e) generate, via the processor, a logical dependency graph data structure by adding logical dependency graph nodes and logical dependency graph edges corresponding to the best matching rule, precedent rules of the best matching rule, and precedent rules of each precedent rule; (f) determine, via the processor, an execution complexity gauge value associated with the generated logical dependency graph data structure; (g) determine, via the processor, a set of distributed worker processes to be utilized to execute the requested calculation; (h) divide, via the processor, the generated logical dependency graph data structure into a set of subgraphs based on the execution complexity gauge value, wherein each worker process in the set of distributed worker processes is assigned a subgraph from the set of subgraphs; (i) initialize, via the processor, each respective worker process in the set of distributed worker processes with the subgraph assigned to the respective worker process; (j) coordinate, via the processor, execution of the requested calculation by worker processes in the set of distributed worker processes, wherein each respective worker process calculates results for logical dependency graph nodes in the subgraph assigned to the respective worker process; and (k)

obtain, via the processor, a computation result of the logical dependency graph node corresponding to the output rule, from the worker process in the set of distributed worker processes that is assigned the subgraph that contains the logical dependency graph node corresponding to the output rule.

BRIEF DESCRIPTION OF THE FIGURES

[0008] The accompanying figures and/or appendices illustrate various exemplary embodiments in accordance with the present disclosure.