TEST DATA GENERATION METHOD AND APPARATUS, MEDIUM, AND COMPUTER SYSTEM

TECHNICAL FIELD

[0001] This specification relates to the field of computer technologies, and in particular, to a test data generation method and apparatus, a medium, and a computer system.

BACKGROUND

[0002] A vulnerability test is a process of detecting a software application, system, program, etc. to find a security vulnerability or bug that may exist in the software applications, systems, programs, etc. A large amount of test data is usually needed, to improve coverage of all branches of running logic of the software program in the test, thereby improving program vulnerability finding efficiency.

SUMMARY

[0003] Embodiments of this specification provide a test data generation method and apparatus, a storage medium, and a computer system, to partially resolve the above-mentioned problems in the conventional technology.

[0004] This specification provides a test data generation method, including: obtaining first test data that triggers a target program abnormality, extracting a first data feature corresponding to the first test data, running a test case corresponding to the first test data, to obtain a first running report, and extracting a first report feature of the first running report; obtaining a plurality of pieces of second test data by changing a value of a field included in the first test data, extracting a plurality of second data features corresponding to the plurality of pieces of second test data, and determining a third data feature from the plurality of second data features based on a first similarity between the first data feature and each second data feature, where the third data feature corresponds to third test data in the plurality of pieces of second test data; and running a test case corresponding to the third test data, to obtain a second running report; extracting a second report feature corresponding

to the second running report; performing feature variation by using the second report feature as an initial value and using the first report feature as a target, to obtain a variant report feature; and determining fourth test data corresponding to the variant report feature, where the third test data and the fourth test data are used to test a target program.

[0005] This specification provides a computer system, including a storage, a processor, and a computer program that is stored in the storage and that is capable of running on the processor. When executing the program, the processor implements the following steps: obtaining first test data that triggers a target program abnormality, extracting a first data feature corresponding to the first test data, running a test case corresponding to the first test data, to obtain a first running report, and extracting a first report feature of the first running report; obtaining a plurality of pieces of second test data by changing a value of a field included in the first test data, extracting a plurality of second data features corresponding to the plurality of pieces of second test data, and determining a third data feature from the plurality of second data features based on a first similarity between the first data feature and each second data feature, where the third data feature corresponds to third test data in the plurality of pieces of second test data; and running a test case corresponding to the third test data, to obtain a second running report; extracting a second report feature corresponding to the second running report; performing feature variation by using the second report feature as an initial value and using the first report feature as a target, to obtain a variant report feature; and determining fourth test data corresponding to the variant report feature, where the third test data and the fourth test data are used to test a target program.

[0006] This specification provides a test data generation apparatus. The apparatus includes: a feature obtaining module, configured to: obtain first test data that triggers a target program abnormality, extract a first data feature corresponding to the first test data, run a test case corresponding to the first test data, to obtain a first running report, and extract a first report feature of the first running report; a first generation module, configured to: obtain a plurality of pieces of second test data by changing a value of a field included in the first test data, extract a plurality of second data features corresponding to the plurality of pieces of second test data, and determine a third data feature from the plurality of second data features based on a first similarity between the first data feature and each second data feature, where the third data feature corresponds to third test data in the plurality of pieces of second test data; and a second generation module, configured to: run a test case corresponding to the third test data, to obtain a second running report; extract a

second report feature corresponding to the second running report; perform feature variation by using the second report feature as an initial value and using the first report feature as a target, to obtain a variant report feature; and determine fourth test data corresponding to the variant report feature, where the third test data and the fourth test data are used to test a target program.

[0007] This specification provides a computer-readable non-volatile storage medium. The storage medium stores a computer program, and when the computer program is executed by a processor, the test data generation method is implemented.

[0008] The above-mentioned at least one technical solution used in embodiments of this specification can achieve at least one of the following beneficial effects: A large amount of high-quality test data for testing the target program can be automatically generated, to improve test coverage of an execution logic branch of the target program, and improve program vulnerability finding efficiency.

BRIEF DESCRIPTION OF DRAWINGS

[0009] The accompanying drawings described here are used to provide a further understanding of this specification and constitute a part of this specification. The example embodiments of this specification and descriptions thereof are used to explain this specification, and do not constitute an improper limitation on this specification. In the accompanying drawings:

[0010] FIG. 1 is a schematic diagram of a test data generation method according to this specification;

[0011] FIG. 2 is a schematic flowchart of a test data generation method according to this specification;

[0012] FIG. 3 is a schematic diagram of feature variation in this specification;

[0013] FIG. 4 is a schematic diagram of a data feature corresponding to a report feature generated via a neural network according to this specification;

[0014] FIG. 5 is a schematic diagram of a training manner of a neural network that generates a data feature based on a report feature according to this specification;

[0015] FIG. 6 is a schematic diagram of generating fused test data according to this specification;

[0016] FIG. 7 is a schematic structural diagram of a test data generation apparatus according

to this specification; and

[0017] FIG. 8 is a schematic structural diagram of a computer system according to an embodiment of this specification.

DESCRIPTION OF EMBODIMENTS

[0018] To make the objectives, technical solutions, and advantages of this specification clearer, the following clearly and comprehensively describes the technical solutions of this specification with reference to specific embodiments and corresponding accompanying drawings of this specification. Clearly, the described embodiments are merely some but not all of embodiments of this specification. All other embodiments obtained by a person of ordinary skill in the art based on the embodiment of this specification without creative efforts shall fall within the protection scope of this specification.

[0019] As described above, a vulnerability test is a process of detecting a software application, system, program, etc. to find a security vulnerability or bug that may exist in the software applications, systems, programs, etc. A large amount of test data is usually needed, to improve coverage of all branches of running logic of the software program in the test, thereby improving program vulnerability finding efficiency. Currently, a test data obtaining manner is to manually construct test data. However, if the test data is only manually constructed, not only a large manual workload needs to be consumed, but also the manually constructed test data is usually limited by experience or a way of thinking of a constructor, making it difficult to construct test data with high coverage of a logic branch of a to-be-tested program. In another manner of obtaining the test data, existing program vulnerability test data is changed based on a preset change rule (for example, a value of some fields in the test data is changed), to generate new test data. However, a software vulnerability is usually hidden in a complex code structure, and the test data is modified based on a preset rule, and is usually modified only in terms of a data mode at a shallow level. For the test data modified in this manner, it is usually difficult to effectively identify and predict a potential software vulnerability in the complex code structure. Therefore, no matter whether the test data is only manually constructed or the test data is obtained by modifying existing test data based on a preset rule, it is difficult to obtain a large amount of test data with high coverage, making it difficult to efficiently find a vulnerability in tested software.

[0020] To resolve the above-mentioned problem, this application provides a test data generation method. FIG. 1 is a schematic diagram of a test data generation method according to this specification. As shown in FIG. 1, test data D1 that has triggered a target program abnormality can be obtained, a feature (for example, A1) of D1 is extracted, a test case corresponding to the test data D1 is run to obtain a running report R1, and a feature (for example, RA1) of R1 is extracted. Values of one to more fields in the test data D1 are changed, for example, to obtain test data D2, a test case corresponding to D2 is run to obtain a running report R2, and a feature (for example, RA2) of R2 is extracted. Feature variation is performed via a neural network by using RA2 as an initial value of R and using RA1 as a target, to obtain a variant report feature RA3. A data feature A3 is obtained based on RA3, and whether a data feature A2 and the data feature A3 are used to test a target program is determined based on a similarity between a data feature A1 and each of the data feature A2 and the data feature A3.

[0021] According to a test data generation method provided in this application, expanded test data is obtained by changing a field. A variant report feature is obtained based on deep learning and by performing variation on a running report feature of the expanded test data, and the further expanded test data is obtained based on the report variation feature. Therefore, first, in the method, the test data can be automatically generated, to reduce dependency on manually generating the test data, and improve efficiency of obtaining the test data. Second, through variation on the running report feature, key features that are not normally identified by human beings can usually be obtained, and high-quality test data is generated based on these key features. Third, a running report usually includes descriptions of a direct running result, a running error, or a thrown abnormality of a test case. Compared with test data obtained only by changing a field value, test data obtained by performing variation on the running report usually can effectively increase coverage of a logic branch of a program, to more effectively identify a potential software vulnerability.

[0022] The following describes in detail the technical solutions provided in the embodiments of this specification with reference to the accompanying drawings.

[0023] FIG. 2 is a schematic flowchart of a test data generation method according to this specification. The method can specifically include the following steps:

[0024] S200: Obtain first test data that triggers a target program abnormality, extract a first data feature corresponding to the first test data, run a test case corresponding to the first test data,

to obtain a first running report, and extract a first report feature of the first running report. In different embodiments, a target program can be used for different specific services or software programs for different specific purposes. This is not limited in this specification.

[0025] Test data, also referred to as test case data, is a set of specific data used to execute a test case for a software program or system. Usually, different test data can be determined based on a test target of different test cases, and is intended to cover various possible execution branches and boundary conditions in software running. Therefore, in different embodiments, the test case data can include different types of data such as a normal value, an abnormal value, a boundary value, or an invalid data of program input data. Further, a vulnerability in software can be effectively found based on high-quality test data.

[0026] In different embodiments, the test data can be in different data formats. In an embodiment, for example, the test data can be in a JavaScript object notation (Yaml) format. Usually, the test data can include a plurality of data fields. In different specific embodiments, the test data can include different specific data fields. For example, in an embodiment, the test data can include a plurality of input data for a target program. In another embodiment, the test data can further include one or more of an execution condition or an expected result of the test case.

[0027] In different embodiments, a data feature corresponding to the test data can be extracted in different manners. In an embodiment, a corresponding multi-dimensional vector feature can be generated based on the test data, and a component of the multi-dimensional vector feature in each dimension can be determined based on a name or a field value of each field in the test data.

[0028] After the first test data is obtained, the test case corresponding to the first test data can be run, to obtain the first running report, and the first report feature of the first running report is extracted. A running report of the test case is a record of an execution status of the test case. In different embodiments, the running report can include different specific content. In an embodiment, for example, the running report can include one or more of a test result (for example, one of "pass", "fail", or "block"), an execution time, an error description (for example, a description of an error or a description of a program thrown abnormality if a test fails), a vulnerability number, a test log, and test environment information.

[0029] In different embodiments, a report feature of the running report can be extracted in different manners. In an embodiment, the report feature of the running report can be extracted via the neural network.

[0030] S202: Obtain a plurality of pieces of second test data by changing a value of a field included in the first test data, extract a plurality of second data features corresponding to the plurality of pieces of second test data, and determine a third data feature from the plurality of second data features based on a first similarity between the first data feature and each second data feature, where the third data feature corresponds to third test data in the plurality of pieces of second test data.

[0031] Specifically, in an embodiment, a value of at least a part of fields included in the first test data can be changed for a plurality of times, to obtain a plurality of pieces of second test data. In this step, a manner of extracting the second data feature can be similar to a manner of extracting the data feature corresponding to the test data in step S200. Details are omitted here for simplicity.

[0032] In different embodiments, the first similarity between the first data feature and each second data feature can be determined in different specific manners. In an embodiment, the first similarity between the first data feature and each second data feature can be determined by calculating a cosine similarity between the first data feature and each second data feature. In different embodiments, the third data feature can be determined from the plurality of second data features in different specific manners. In an embodiment, the plurality of second data features can be sorted based on a value of the similarity (the first similarity) between the first data feature and each of the plurality of second data features, and a second data feature sorted in front, for example, accounting for a first predetermined proportion can be determined as the third data feature. In different specific embodiments, the first predetermined proportion can have different values. In a specific embodiment, for example, the second data feature sorted in front, for example, accounting for 10% can be determined as the third data feature.

[0033] S204: Run a test case corresponding to the third test data, to obtain a second running report; extract a second report feature corresponding to the second running report; perform feature variation by using the second report feature as an initial value and using the first report feature as a target, to obtain a variant report feature; and determine fourth test data corresponding to the variant report feature, where the third test data and the fourth test data are used to test the target program. In this step, a manner of extracting the second report feature can be similar to a manner of extracting the report feature corresponding to the running report in S200. Details are omitted here for simplicity.

[0034] In different embodiments, feature variation can be performed in different specific

manners. In an embodiment, feature variation can be performed based on a fast gradient descent algorithm by using the second report feature as the initial value and using the first report feature as the target, to obtain the variant report feature. Specifically, FIG. 3 is a schematic diagram of feature variation according to this specification. As shown in FIG. 3, the first report feature can be represented as, for example, a vector $\vec{V}$, and the second report feature can be represented as, for example, a vector $\overrightarrow{V'}$. $\overrightarrow{V'}$ and $\vec{V}$ can be input into the neural network, and $\overrightarrow{V'}$ is enabled to vary in a direction of approaching $\vec{V}$ based on the fast gradient descent algorithm by using $\overrightarrow{V'}$ as an initial value of feature variation, to obtain a new variant report feature (which is represented as $\overrightarrow{V^*}$)

[0035] FIG. 4 is a schematic diagram of a data feature corresponding to a report feature generated via a neural network according to this specification. As shown in FIG. 4, in an embodiment, the variant report feature (report feature RA3) can be input into a pre-trained first neural network (neural network N2), to obtain a fourth data feature (data feature A3). The first neural network is configured to: output a feature of the test data based on a feature of the running report; and determine the fourth test data based on the fourth data feature. FIG. 5 is a schematic diagram of a training manner of a neural network that generates a data feature based on a report feature according to this specification. As shown in FIG. 5, in a specific embodiment, in pre-training of the first neural network, the first report feature (report feature RA1) and the second report feature (report feature RA2) can be used as training samples, and the first data feature (data feature A1) and the second data feature (data feature A2) can be used as training labels corresponding to the training samples. In an embodiment, a name or a field of each field in the fourth test data can be determined based on a value of a component in each dimension in a multi-dimensional vector corresponding to the fourth data feature.

[0036] To further improve quality of test data used to test the target program, in an embodiment, a third similarity between the first data feature and the variant data feature can be determined. If the third similarity reaches a predetermined threshold, the fourth test data is used to test the target program. In different specific embodiments, the predetermined threshold can have different specific values.

[0037] To further expand an amount of test data, in an embodiment, preferred test data and secondary test data can be further determined from the third test data and the fourth test data, and

the preferred test data and the secondary test data are combined, to obtain fifth test data. The fifth test data is used to test the target program. In different embodiments, the preferred test data and the secondary test data can be determined in different manners. In an embodiment, for example, test data whose amount accounts for the second predetermined proportion in the third test data and the fourth test data can be used as preferred test data, and remaining test data is used as secondary test data. A similarity between a data feature of each piece of preferred test data and the first data feature is higher than a similarity between a data feature of each piece of secondary test data and the first data feature. In different specific embodiments, the second predetermined proportion can have different values. In a specific embodiment, for example, test data whose amount accounts for, for example, 20% in the third test data and the fourth test data can be used as preferred test data, and remaining test data is used as secondary test data.

[0038] In different embodiments, the preferred test data and the secondary test data can be combined in different specific manners. In a specific embodiment, a value of a first field and a value of a second field in the fifth test data can be determined based on the value of the first field in the preferred test data and the value of the second field in the secondary test data. FIG. 6 is a schematic diagram of generating fused test data according to this specification. As shown in FIG. 6, fields of the test data include, for example, a field C1, a field C2, a field C3, and a field C4. The first field can include the field C1 and the field C2, and the second field can include the field C3 and the field C4. A value of the field C1, a value of the field C2, a value of the field C3, and a value of the field C4 in the fused test data (fifth test data) D5 can be determined based on the value of the field C1 and the value of the field C2 in the preferred test data D3 and the value of the field C3 and the value of the field C4 in the secondary test data D4.

[0039] Similarly, to further improve quality of the test data used to test the target program, in another specific embodiment, a second similarity between a data feature corresponding to the fifth test data and the first data feature is determined. If the second similarity reaches a predetermined threshold, the fifth test data is used to test the target program.

[0040] The test data generation method provided in the embodiments of this specification are provided above. Based on the same idea, this specification further provides a corresponding test data generation apparatus. As shown in FIG. 7, the test data generation apparatus includes: a feature obtaining module 702, configured to: obtain first test data that triggers a target program abnormality, extract a first data feature corresponding to the first test data, run a test case

corresponding to the first test data, to obtain a first running report, and extract a first report feature of the first running report; a first generation module 704, configured to: obtain a plurality of pieces of second test data by changing a value of a field included in the first test data, extract a plurality of second data features corresponding to the plurality of pieces of second test data, and determine a third data feature from the plurality of second data features based on a first similarity between the first data feature and each second data feature, where the third data feature corresponds to third test data in the plurality of pieces of second test data; and a second generation module 706, configured to: run a test case corresponding to the third test data, to obtain a second running report; extract a second report feature corresponding to the second running report; perform feature variation by using the second report feature as an initial value and using the first report feature as a target, to obtain a variant report feature; and determine fourth test data corresponding to the variant report feature, where the third test data and the fourth test data are used to test a target program.

[0041] For ease of description, the above-mentioned apparatus is described by dividing the apparatus into various modules based on functions. Certainly, during implementation of this specification, functions of all modules can be implemented in one or more pieces of software and/or hardware.

[0042] The specification also provides a computer-readable non-transitory storage medium, where the storage medium stores a computer program, which, when executed by a processor, can be used to perform one or more steps of one or more methods described or illustrated herein or provides functionality described or illustrated herein. Herein, the computer-readable non-transitory storage medium or media may include one or more semiconductor-based or other integrated circuits (ICs) (such, as for example, field-programmable gate arrays (FPGAs) or application-specific ICs (ASICs)), hard disk drives (HDDs), hybrid hard drives (HHDs), optical discs, optical disc drives (ODDs), magneto-optical discs, magneto-optical drives, floppy diskettes, floppy disk drives (FDDs), magnetic tapes, solid-state drives (SSDs), RAM-drives, SECURE DIGITAL cards or drives, any other suitable computer-readable non-transitory storage media, or any suitable combination of two or more of these, where appropriate. A computer-readable non-transitory storage medium may be volatile, non-volatile, or a combination of volatile and non-volatile, where appropriate.

[0043] The specification also provides a computer system. FIG. 8 illustrates an example

computer system 800. In particular embodiments, one or more computer systems 800 perform one or more steps of one or more methods described or illustrated herein. In particular embodiments, one or more computer systems 800 provide functionality described or illustrated herein. In particular embodiments, software running on one or more computer systems 800 performs one or more steps of one or more methods described or illustrated herein or provides functionality described or illustrated herein. Particular embodiments include one or more portions of one or more computer systems 800. Herein, reference to a computer system may encompass a computing device, and vice versa, where appropriate. Moreover, reference to a computer system may encompass one or more computer systems, where appropriate.

[0044] This disclosure contemplates any suitable number of computer systems 800. This disclosure contemplates computer system 800 taking any suitable physical form. As example and not by way of limitation, computer system 800 may be an embedded computer system, a system-on-chip (SOC), a single-board computer system (SBC) (such as, for example, a computer-on-module (COM) or system-on-module (SOM)), a desktop computer system, a laptop or notebook computer system, an interactive kiosk, a mainframe, a mesh of computer systems, a mobile telephone, a personal digital assistant (PDA), a server, a tablet computer system, or a combination of two or more of these. Where appropriate, computer system 800 may include one or more computer systems 800; be unitary or distributed; span multiple locations; span multiple machines; span multiple data centers; or reside in a cloud, which may include one or more cloud components in one or more networks. Where appropriate, one or more computer systems 800 may perform without substantial spatial or temporal limitation one or more steps of one or more methods described or illustrated herein. As an example and not by way of limitation, one or more computer systems 800 may perform in real time or in batch mode one or more steps of one or more methods described or illustrated herein. One or more computer systems 800 may perform at different times or at different locations one or more steps of one or more methods described or illustrated herein, where appropriate.

[0045] In particular embodiments, computer system 800 includes a processor 802, memory 804, storage 806, an input/output (I/O) interface 808, a communication interface 810, and a bus 812. Although this disclosure describes and illustrates a particular computer system having a particular number of particular components in a particular arrangement, this disclosure contemplates any suitable computer system having any suitable number of any suitable

components in any suitable arrangement.

[0046] In particular embodiments, processor 802 includes hardware for executing instructions, such as those making up a computer program. As an example and not by way of limitation, to execute instructions, processor 802 may retrieve (or fetch) the instructions from an internal register, an internal cache, memory 804, or storage 806; decode and execute them; and then write one or more results to an internal register, an internal cache, memory 804, or storage 806. In particular embodiments, processor 802 may include one or more internal caches for data, instructions, or addresses. This disclosure contemplates processor 802 including any suitable number of any suitable internal caches, where appropriate. As an example and not by way of limitation, processor 802 may include one or more instruction caches, one or more data caches, and one or more translation lookaside buffers (TLB s). Instructions in the instruction caches may be copies of instructions in memory 804 or storage 806, and the instruction caches may speed up retrieval of those instructions by processor 802. Data in the data caches may be copies of data of 80 in memory 804 or storage 806 for instructions executing at processor 802 to operate on; the results of previous instructions executed at processor 802 for access by subsequent instructions executing at processor 802 or for writing to memory 804 or storage 806; or other suitable data. The data caches may speed up read or write operations by processor 802. The TLBs may speed up virtual-address translation for processor 802. In particular embodiments, processor 802 may include one or more internal registers for data, instructions, or addresses. This disclosure contemplates processor 802 including any suitable number of any suitable internal registers, where appropriate. Where appropriate, processor 802 may include one or more arithmetic logic units (ALUs); be a multi-core processor; or include one or more processors 802. Although this disclosure describes and illustrates a particular processor, this disclosure contemplates any suitable processor.

[0047] In particular embodiments, memory 804 includes main memory for storing instructions for processor 802 to execute or data for processor 802 to operate on. As an example and not by way of limitation, computer system 800 may load instructions from storage 806 or another source (such as, for example, another computer system 800) to memory 804. Processor 802 may then load the instructions from memory 804 to an internal register or internal cache. To execute the instructions, processor 802 may retrieve the instructions from the internal register or internal cache and decode them. During or after execution of the instructions, processor 802 may write one or more results (which may be intermediate or final results) to the internal register or internal cache.

Processor 802 may then write one or more of those results to memory 804. In particular embodiments, processor 802 executes only instructions in one or more internal registers or internal caches or in memory 804 (as opposed to storage 806 or elsewhere) and operates only on data in one or more internal registers or internal caches or in memory 804 (as opposed to storage 806 or elsewhere). One or more memory buses (which may each include an address bus and a data bus) may couple processor 802 to memory 804. Bus 812 may include one or more memory buses, as described below. In particular embodiments, one or more memory management units (MMUs) reside between processor 802 and memory 804 and facilitate accesses to memory 804 requested by processor 802. In particular embodiments, memory 804 includes random access memory (RAM). This RAM may be volatile memory, where appropriate. Where appropriate, this RAM may be dynamic RAM (DRAM) or static RAM (SRAM). Moreover, where appropriate, this RAM may be single-ported or multi-ported RAM. This disclosure contemplates any suitable RAM. Memory 804 may include one or more memories 804, where appropriate. Although this disclosure describes and illustrates particular memory, this disclosure contemplates any suitable memory.

[0048] In particular embodiments, storage 806 includes mass storage for data or instructions. As an example and not by way of limitation, storage 806 may include a hard disk drive (HDD), a floppy disk drive, flash memory, an optical disc, a magneto-optical disc, magnetic tape, or a Universal Serial Bus (USB) drive or a combination of two or more of these. Storage 806 may include removable or non-removable (or fixed) media, where appropriate. Storage 806 may be internal or external to computer system 800, where appropriate. In particular embodiments, storage 806 is non-volatile, solid-state memory. In particular embodiments, storage 806 includes read-only memory (ROM). Where appropriate, this ROM may be mask-programmed ROM, programmable ROM (PROM), erasable PROM (EPROM), electrically erasable PROM (EEPROM), electrically alterable ROM (EAROM), or flash memory or a combination of two or more of these. This disclosure contemplates mass storage 806 taking any suitable physical form. Storage 806 may include one or more storage control units facilitating communication between processor 802 and storage 806, where appropriate. Where appropriate, storage 806 may include one or more storages 806. Although this disclosure describes and illustrates particular storage, this disclosure contemplates any suitable storage.

[0049] In particular embodiments, I/O interface 808 includes hardware, software, or both, providing one or more interfaces for communication between computer system 800 and one or

more I/O devices. Computer system 800 may include one or more of these I/O devices, where appropriate. One or more of these I/O devices may enable communication between a person and computer system 800. As an example and not by way of limitation, an I/O device may include a keyboard, keypad, microphone, monitor, mouse, printer, scanner, speaker, still camera, stylus, tablet, touch screen, trackball, video camera, another suitable I/O device or a combination of two or more of these. An I/O device may include one or more sensors. This disclosure contemplates any suitable I/O devices and any suitable I/O interfaces 808 for them. Where appropriate, I/O interface 808 may include one or more device or software drivers enabling processor 802 to drive one or more of these I/O devices. I/O interface 808 may include one or more I/O interfaces 808, where appropriate. Although this disclosure describes and illustrates a particular I/O interface, this disclosure contemplates any suitable I/O interface.

[0050] In particular embodiments, communication interface 810 includes hardware, software, or both providing one or more interfaces for communication (such as, for example, packet-based communication) between computer system 800 and one or more other computer systems 800 or one or more networks. As an example and not by way of limitation, communication interface 810 may include a network interface controller (NIC) or network adapter for communicating with an Ethernet or other wire-based network or a wireless NIC (WNIC) or wireless adapter for communicating with a wireless network, such as a WI-FI network. This disclosure contemplates any suitable network and any suitable communication interface 810 for it. As an example and not by way of limitation, computer system 800 may communicate with an ad hoc network, a personal area network (PAN), a local area network (LAN), a wide area network (WAN), a metropolitan area network (MAN), or one or more portions of the Internet or a combination of two or more of these. One or more portions of one or more of these networks may be wired or wireless. As an example, computer system 800 may communicate with a wireless PAN (WPAN) (such as, for example, a BLUETOOTH WPAN), a WI-FI network, a WI-MAX network, a cellular telephone network (such as, for example, a Global System for Mobile Communications (GSM) network), or other suitable wireless network or a combination of two or more of these. Computer system 800 may include any suitable communication interface 810 for any of these networks, where appropriate. Communication interface 810 may include one or more communication interfaces 810, where appropriate. Although this disclosure describes and illustrates a particular communication interface, this disclosure contemplates any suitable communication interface.

[0051] In particular embodiments, bus 812 includes hardware, software, or both coupling components of computer system 800 to each other. As an example and not by way of limitation, bus 812 may include an Accelerated Graphics Port (AGP) or other graphics bus, an Enhanced Industry Standard Architecture (EISA) bus, a front-side bus (FSB), a HYPERTRANSPORT (HT) interconnect, an Industry Standard Architecture (ISA) bus, an INFINIBAND interconnect, a low-pin-count (LPC) bus, a memory bus, a Micro Channel Architecture (MCA) bus, a Peripheral Component Interconnect (PCI) bus, a PCI-Express (PCIe) bus, a serial advanced technology attachment (SATA) bus, a Video Electronics Standards Association local (VLB) bus, or another suitable bus or a combination of two or more of these. Bus 812 may include one or more buses 812, where appropriate. Although this disclosure describes and illustrates a particular bus, this disclosure contemplates any suitable bus or interconnect.

[0052] It should be further noted that, the terms "include", "comprise", or any other variant thereof are intended to cover a non-exclusive inclusion, so that a process, a method, a product or a device that includes a list of elements not only includes those elements but also includes other elements which are not expressly listed, or further includes elements inherent to such process, method, product, or device. Without more constraints, an element preceded by "includes a …" does not preclude the existence of additional identical elements in the process, method, product, or device that includes the element.

[0053] The embodiments of this specification are described in a progressive way. For same or similar parts in the embodiments, references can be made to each other. Each embodiment focuses on a difference from another embodiment. Particularly, some system embodiments are briefly described because they are basically similar to some method embodiments. For related parts, references can be made to related descriptions in some method embodiments.

[0054] The above-mentioned descriptions are merely embodiments of this specification, and are not intended to limit this specification. A person skilled in the art can make various changes and changes to this specification. Any modification, equivalent replacement, improvement, etc. made without departing from the spirit and principle of this specification shall fall within the scope of the claims in this application.

CLAIMS

What is claimed is:

1. A test data generation method, comprising:

obtaining first test data that triggers a target program abnormality, extracting a first data feature corresponding to the first test data, running a test case corresponding to the first test data, to obtain a first running report, and extracting a first report feature of the first running report;

obtaining a plurality of pieces of second test data by changing a value of a field comprised in the first test data, extracting a plurality of second data features corresponding to the plurality of pieces of second test data, and determining a third data feature from the plurality of second data features based on a first similarity between the first data feature and each second data feature, wherein the third data feature corresponds to third test data in the plurality of pieces of second test data; and

running a test case corresponding to the third test data, to obtain a second running report; extracting a second report feature corresponding to the second running report; performing feature variation by using the second report feature as an initial value and using the first report feature as a target, to obtain a variant report feature; and determining fourth test data corresponding to the variant report feature, wherein the third test data and the fourth test data are used to test a target program.

2. The method according to claim 1, further comprising:

determining preferred test data and secondary test data from the third test data and the fourth test data, and combining the preferred test data and the secondary test data, to obtain fifth test data, wherein the fifth test data is used to test the target program.

3. The method according to claim 2, wherein the combining the preferred test data and the secondary test data, to obtain fifth test data comprises:

determining a value of a first field and a value of a second field in the fifth test data based on the value of the first field in the preferred test data and the value of the second field in the secondary test data.

4. The method according to claim 2, wherein that the fifth test data is used to test the target program comprises:

determining a second similarity between a data feature corresponding to the fifth test data and

the first data feature, wherein if the second similarity reaches a predetermined threshold, the fifth test data is used to test the target program.

5. The method according to claim 1, wherein the determining fourth test data corresponding to the variant report feature comprises:

inputting the variant report feature into a pre-trained first neural network, to obtain a fourth data feature, wherein the first neural network is configured to output a feature of test data based on a feature of a running report; and

determining the fourth test data based on the fourth data feature.

6. The method according to claim 5, wherein in pre-training of the first neural network, the first report feature and the second report feature are used as training samples, and the first data feature and the third data feature are used as training labels corresponding to the training samples.

7. The method according to claim 1, wherein the performing feature variation by using the second report feature as an initial value and using the first report feature as a target, to obtain a variant report feature comprises:

performing, based on a fast gradient descent algorithm, feature variation by using the second report feature as the initial value and using the first report feature as the target, to obtain the variant report feature.

8. The method according to claim 1, wherein that the fourth test data is used to test the target program comprises:

determining a third similarity between the first data feature and a variant data feature, wherein if the third similarity reaches a predetermined threshold, the fourth test data is used to test the target program.

9. A computer system, comprising a storage, a processor, and a computer program stored in the storage and running on the processor, wherein when executing the program, the processor implements the following steps:

obtaining first test data that triggers a target program abnormality, extracting a first data feature corresponding to the first test data, running a test case corresponding to the first test data, to obtain a first running report, and extracting a first report feature of the first running report;

obtaining a plurality of pieces of second test data by changing a value of a field comprised in the first test data, extracting a plurality of second data features corresponding to the plurality of pieces of second test data, and determining a third data feature from the plurality of second data

features based on a first similarity between the first data feature and each second data feature, wherein the third data feature corresponds to third test data in the plurality of pieces of second test data; and

running a test case corresponding to the third test data, to obtain a second running report; extracting a second report feature corresponding to the second running report; performing feature variation by using the second report feature as an initial value and using the first report feature as a target, to obtain a variant report feature; and determining fourth test data corresponding to the variant report feature, wherein the third test data and the fourth test data are used to test a target program.

10. The computer system according to claim 9, wherein the steps further comprise:

determining preferred test data and secondary test data from the third test data and the fourth test data, and combining the preferred test data and the secondary test data, to obtain fifth test data, wherein the fifth test data is used to test the target program.

11. The computer system according to claim 10, wherein the combining the preferred test data and the secondary test data, to obtain fifth test data comprises:

determining a value of a first field and a value of a second field in the fifth test data based on the value of the first field in the preferred test data and the value of the second field in the secondary test data.

12. The computer system according to claim 10, wherein that the fifth test data is used to test the target program comprises:

determining a second similarity between a data feature corresponding to the fifth test data and the first data feature, wherein if the second similarity reaches a predetermined threshold, the fifth test data is used to test the target program.

13. The computer system according to claim 9, wherein the determining fourth test data corresponding to the variant report feature comprises:

inputting the variant report feature into a pre-trained first neural network, to obtain a fourth data feature, wherein the first neural network is configured to output a feature of test data based on a feature of a running report; and

determining the fourth test data based on the fourth data feature.

14. The computer system according to claim 13, wherein in pre-training of the first neural network, the first report feature and the second report feature are used as training samples, and the

first data feature and the third data feature are used as training labels corresponding to the training samples.

15. The computer system according to claim 9, wherein the performing feature variation by using the second report feature as an initial value and using the first report feature as a target, to obtain a variant report feature comprises:

performing, based on a fast gradient descent algorithm, feature variation by using the second report feature as the initial value and using the first report feature as the target, to obtain the variant report feature.

16. The computer system according to claim 9, wherein that the fourth test data is used to test the target program comprises:

determining a third similarity between the first data feature and a variant data feature, wherein if the third similarity reaches a predetermined threshold, the fourth test data is used to test the target program.

17. A test data generation apparatus, wherein the apparatus comprises:

a feature obtaining module, configured to: obtain first test data that triggers a target program abnormality, extract a first data feature corresponding to the first test data, run a test case corresponding to the first test data, to obtain a first running report, and extract a first report feature of the first running report;

a first generation module, configured to: obtain a plurality of pieces of second test data by changing a value of a field comprised in the first test data, extract a plurality of second data features corresponding to the plurality of pieces of second test data, and determine a third data feature from the plurality of second data features based on a first similarity between the first data feature and each second data feature, wherein the third data feature corresponds to third test data in the plurality of pieces of second test data; and

a second generation module, configured to: run a test case corresponding to the third test data, to obtain a second running report; extract a second report feature corresponding to the second running report; perform feature variation by using the second report feature as an initial value and using the first report feature as a target, to obtain a variant report feature; and determine fourth test data corresponding to the variant report feature, wherein the third test data and the fourth test data are used to test a target program.

18. A computer-readable non-volatile storage medium, wherein the storage medium stores a

computer program, and when the computer program is executed by a processor, the method according to any one of claims 1 to 8 is implemented.