



(10) **DE 10 2012 216 841 A1** 2013.04.04

(12)

Offenlegungsschrift

(21) Aktenzeichen: **10 2012 216 841.9**

(22) Anmeldetag: **20.09.2012**

(43) Offenlegungstag: **04.04.2013**

(51) Int Cl.: **G06F 11/28 (2012.01)**

G06F 11/36 (2012.01)

G06F 17/50 (2012.01)

(30) Unionspriorität:
201110293769.3 29.09.2011 CN

(72) Erfinder:
Tang, Wen, Beijing, CN

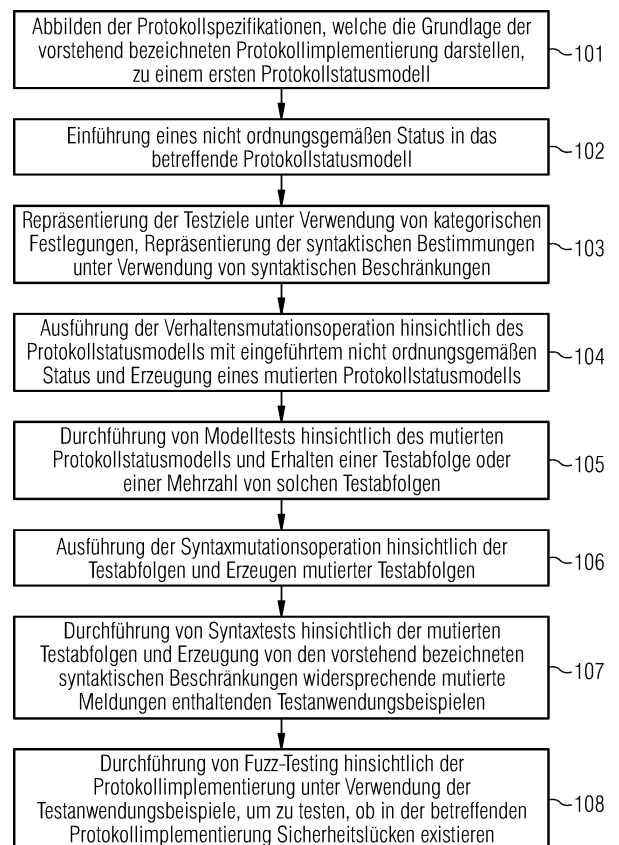
(71) Anmelder:
Siemens Aktiengesellschaft, 80333, München, DE

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **Verfahren und Vorrichtung zur Durchführung von Sicherheitstests hinsichtlich von Protokollimplementierungen**

(57) Zusammenfassung: Die vorliegende Erfindung betrifft ein Verfahren und eine Vorrichtung zur Durchführung von Sicherheitstests hinsichtlich von Protokollimplementierungen, wobei das vorstehend bezeichnete Verfahren beinhaltet: Abbilden der Protokollspezifikationen zu einem ersten Protokollstatusmodell; Einführen eines nicht ordnungsgemäßen Status zur Erzeugung eines zweiten Protokollstatusmodells in das vorstehend bezeichnete erste Protokollstatusmodell; unter Verwendung kategorischer Festlegungen, Repräsentieren von Testzielen des betreffenden ersten Protokollstatusmodells/des betreffenden zweiten Protokollstatusmodells, unter Verwendung syntaktischer Beschränkungen, Repräsentieren der syntaktischen Bestimmungen für ordnungsgemäße Eingabemeldungen des betreffenden Modells; hinsichtlich des zweiten Protokollstatusmodells, Ausführen der Verhaltensmutationsoperation, zur Erzeugung eines mutierten Protokollstatusmodells; und Durchführen des Modelltests, wobei die erhaltenen, den kategorischen Festlegungen widersprechenden Gegenbeispiele zu Testabfolgen abgebildet werden; hinsichtlich der Testabfolgen Ausführen der Syntaxmutationsoperation, um mutierte Testmeldungen enthaltende Testabfolgen zu erzeugen, wobei darunter befindliche, gültige mutierte Meldungen enthaltende Testabfolgen als Testanwendungsbeispiele verwendet werden; unter Verwendung der entsprechenden Testanwendungsbeispiele, Durchführen von Fuzz-Testing hinsichtlich der Protokollimplementierung, um in der Protokollimplementierung existierende Sicherheitslücken zu testen. Durch Verwendung der praktischen Ausführungsbeispiele gemäß der vorliegenden Erfindung kann mit hoher Effizienz und zu geringen Kosten die Durchführung von Sicherheitstests hinsichtlich von Protokollimplementierungen vorgenommen werden.



Beschreibung

Technisches Gebiet

[0001] Die vorliegende Erfindung betrifft das technische Gebiet der Informationssicherheit (Information Security), insbesondere ein Verfahren und eine Vorrichtung zur Durchführung von Sicherheitstest hinsichtlich von Protokollimplementierungen.

Stand der Technik

[0002] Die Technik der Computerinformationssicherheit ist eine umfassende technische Wissenschaft, welche mehrere einzelne technische Bereiche wie beispielsweise die Anwendung von Codierungen, die Wahrung von Informationssicherheit, die Handhabung bei Datenschäden und entsprechende Datenwiederherstellung, die Durchführung von Betriebssystemwartung usw. umfasst. Hierbei wird die Informationssicherheitstechnik hauptsächlich zur Verhinderung von Systemlücken eingesetzt, um das Eindringen externer Hacker zu verhindern, um Schäden durch Computerviren zu verhindern und um eine effektive Kontrolle gegen zweifelhafte Zugriffe auszuüben.

[0003] Im Zuge der Entwicklung des Internets besteht für die verschiedenen Computersysteme die dringende Erfordernis, eine Intensivierung der Informationssicherheitstechnik vorzunehmen. Als Beispiel sollen Systeme der Informations- und Kommunikationstechnologie (ICT Information Communication Technology) und der Kommunikationstechnologie (Communication Technology) dienen. Bei der Informations- und Kommunikationstechnologie ICT handelt es sich um eine an Kunden erbrachte Dienstleistung, welche eine Verbindung und Verschmelzung von Informationstechnologie (IT, Information Technology) mit Kommunikationstechnologie (CT, Communication Technology) darstellt.

[0004] Kommunikationsindustrie, elektronische Informationsindustrie, Internet und Medienbranche verschmelzen im Bereich der Informations- und Kommunikationstechnologie ICT, so dass für die Kunden die Bereitstellung umfassend integrierter Dienstleistungen ermöglicht wird, einschließlich, integrierter Dienste, externer Dienstleistungen, spezieller Dienstleistungen, geistiger Dienstleistungen, Dienstleistungen von Software-Entwicklung usw. Allerdings kam es im Zuge der Entwicklung der Informations- und Kommunikationstechnologie ICT auch zu einer signifikanten Erhöhung der Anforderungen hinsichtlich der entsprechenden Basisstruktur. Beispielhaft führen die Erhöhung der Konnektivität zwischen verschiedenen Projektbeteiligten (beispielsweise Gemeinschaftsunternehmen, Partnerunternehmen, externe Dienstleister), die Erhöhung der Konnektivität zwischen intelligenten Vorrichtungen, die Stärkung der externen Konnektivität und das rasch wachsende Internet zu vermehrten Vorfällen des Eindringens von Hackern oder von Schadsoftware. Dies alles bewirkt Angriffe auf die Basisstruktur von Systemen der Informations- und Kommunikationstechnologie ICT und stellt eine diesbezüglich signifikant erhöhte Bedrohungslage dar. Aus diesem Grund besteht ein außerordentlicher Bedarf an einer Intensivierung der Informationssicherheitstechnik von Systemen der Informations- und Kommunikationstechnologie ICT, um hinsichtlich von Sicherheitslücken und/oder selbst hervorgerufenen Störungen die Sicherheit der Kommunikations- und Informationsübertragung von Systemen der Informations- und Kommunikationstechnologie ICT zu gewährleisten.

[0005] Bei der Informationssicherheitstechnik besteht eine bedeutende Thematik darin, Sicherheitstests von Protokollen, welche auf Protokollimplementierung basieren, durchzuführen. Derzeit vorhandene verschiedene Verfahren zur Durchführung von Sicherheitstests für Protokolle sind: Informelle Verifikation (Informal Verification), White-box-Tests (White-box), Blackbox-Tests (Black-box) usw. Hierbei ist für die informelle Verifikation und die White-box-Tests der Quellcode (Source Code) des Softwareprogramms der im Test befindlichen Implementierung (IUT, Implementation Under Test) erforderlich. Daher ist der Anwendungsbereich dieser Verfahren außerordentlich eingeschränkt. Andererseits basiert der Black-box-Test auf einer externen Beschreibung des Protokolls. Dies bedeutet, dass der Test der Protokollspezifikationen (Specifications) nicht auf dem Quellcode des Computerprogramms beruht. Außerdem kann die Implementierung des Black-box-Tests unabhängig von dem Computerprogramm der Protokollimplementierung sein, so dass die Testergebnisse für sämtliche Implementierungen der Protokollspezifikationen verwendet werden können. Daher kann der Black-box-Test umfassend für verschiedene Testsituationen verwendet werden. Allerdings sind die Kosten der Implementierung von Black-box-Tests außerordentlich hoch, bedeuten einen enormen Zeitaufwand und sind schwer zu optimieren oder zu verbessern.

[0006] Aus diesem Grund besteht auf dem Gebiet der Informationssicherheitstechnik gegenwärtig Bedarf an der Entwicklung einer Technik zur Durchführung von Sicherheitstests für Protokolle, bei welcher der Quell-

code des Computerprogramms der Protokollimplementierung nicht erforderlich ist und wobei eine hohe Ausführungseffizienz bei gleichzeitig niedrigen Kosten der praktischen Umsetzung gewährleistet wird.

Inhalt der Erfindung

[0007] Zwecks Lösung der vorstehend bezeichneten technischen Probleme erfolgt durch die praktischen Ausführungsbeispiele der vorliegenden Erfindung die Bereitstellung von Verfahren und Vorrichtung zur Durchführung von Sicherheitstests hinsichtlich von Protokollimplementierungen. Hierbei ist der Quellcode des Computerprogramms der zu testenden Protokollimplementierung nicht erforderlich, um den Sicherheitstest der Protokollimplementierungen zu verwirklichen. Die vorliegende Erfindung bietet eine hohe Ausführungseffizienz bei geringen Kosten der praktischen Umsetzung.

[0008] Durch die praktischen Ausführungsbeispiele der vorliegenden Erfindung erfolgt die Bereitstellung eines Verfahrens zur Durchführung von Sicherheitstests hinsichtlich von Protokollimplementierungen, wobei das betreffende Verfahren Folgendes beinhaltet:

Abbilden der Protokollspezifikationen, welche die Grundlage der vorstehend bezeichneten Protokollimplementierung darstellen, zu einem ersten Protokollstatusmodell;

Einführen eines nicht ordnungsgemäßen Status zur Erzeugung eines zweiten Protokollstatusmodells in das vorstehend bezeichnete erste Protokollstatusmodell, wobei der vorstehend bezeichnete nicht ordnungsgemäße Status in der vorstehend bezeichneten Protokollimplementierung existierenden Sicherheitsstörungen entspricht;

unter Verwendung kategorischer Festlegungen, Repräsentieren von Testzielen des vorstehend bezeichneten ersten Protokollstatusmodells und/oder des vorstehend bezeichneten zweiten Protokollstatusmodells;

unter Verwendung syntaktischer Beschränkungen, Repräsentieren der syntaktischen Bestimmungen für ordnungsgemäße Eingabemeldungen von vorstehend bezeichnetem ersten Protokollstatusmodell und/oder vorstehend bezeichnetem zweiten Protokollstatusmodell;

hinsichtlich des vorstehend bezeichneten zweiten Protokollstatusmodells, Ausführen der Verhaltensmutationsoperation, um ein mutiertes Protokollstatusmodell zu erzeugen;

entsprechend den vorstehend bezeichneten kategorischen Festlegungen Durchführen des Modelltests hinsichtlich des vorstehend bezeichneten mutierten Protokollstatusmodells und Erzeugen eines den vorstehend bezeichneten kategorischen Festlegungen widersprechenden Gegenbeispiels oder einer Mehrzahl von solchen Gegenbeispielen, wobei die vorstehend bezeichneten Gegenbeispiele zu Testabfolgen abgebildet werden;

hinsichtlich der vorstehend bezeichneten Testabfolgen, Ausführen der Syntaxmutationsoperation, um mutierte Meldungen enthaltende Testabfolgen zu erzeugen, wobei jede Testabfolge eine mutierte Testabfolge erzeugt, welche eine mutierte Meldung oder eine Mehrzahl von solchen mutierten Meldungen enthält;

entsprechend den vorstehend bezeichneten syntaktischen Beschränkungen, Durchführen des Syntaxtests hinsichtlich der in den vorstehend bezeichneten mutierten Testabfolgen befindlichen mutierten Meldungen, um in den vorstehend bezeichneten mutierten Meldungen enthaltene und den vorstehend bezeichneten syntaktischen Beschränkungen widersprechende gültige mutierte Meldungen zu erhalten, wobei die vorstehend bezeichnete gültige mutierte Meldungen enthaltenden vorstehend bezeichneten mutierten Testabfolgen als Testanwendungsbeispiele verwendet werden; und

unter Verwendung der vorstehend bezeichneten Testanwendungsbeispiele, Durchführen von Fuzz-Testing hinsichtlich der Protokollimplementierung, um zu testen, ob in der vorstehend bezeichneten Protokollimplementierung einem der vorstehend bezeichneten Testanwendungsbeispiele oder einer Mehrzahl von solchen Testanwendungsbeispielen entsprechende Sicherheitslücken existieren.

[0009] Hierbei wird vorgesehen, dass es sich bei dem vorstehend bezeichneten ersten Protokollstatusmodell um einen parameterisierten erweiterten endlichen Zustandsautomaten PEFSM beziehungsweise um ein Input Output Labeled Transition System IOLTS handelt.

[0010] Hierbei wird vorgesehen, dass die vorstehend bezeichnete Verwendung von kategorischen Festlegungen zur Repräsentation der Testziele von vorstehend bezeichnetem ersten Protokollstatusmodell und/oder vorstehend bezeichnetem zweiten Protokollstatusmodell Folgendes beinhaltet:

basierend auf den gegebenen Testzielen, Definieren von kategorischen Festlegungen; und/oder

entsprechend von aus dem Bereich der Sicherheitstests stammendem Wissen, vorab Definieren von kategorischen Festlegungen; und/oder

aus dem vorstehend bezeichneten ersten Protokollstatusmodell, Reflexieren von kategorischen Festlegungen.

[0011] Hierbei wird vorgesehen, dass die vorstehend bezeichnete Verwendung syntaktischer Beschränkungen Folgendes beinhaltet:

entsprechend den Protokollspezifikationen, Definieren von syntaktischen Beschränkungen; und/oder aus den ordnungsgemäßen Eingabemeldungen des vorstehend bezeichneten ersten Protokollstatusmodells, Reflexieren von syntaktischen Beschränkungen.

[0012] Hierbei wird vorgesehen, dass die vorstehend bezeichnete Ausführung der Verhaltensmutationsoperation hinsichtlich des vorstehend bezeichneten zweiten Protokollstatusmodells Folgendes beinhaltet:

unter Verwendung von Verhaltensmutationsoperatoren Ausführen der Verhaltensmutationsoperation hinsichtlich der Übertragung sowie hinsichtlich der logischen Operationen bei der Übertragung in dem vorstehend bezeichneten zweiten Protokollstatusmodell.

[0013] Hierbei wird vorgesehen, dass die vorstehend bezeichnete Ausführung der Verhaltensmutationsoperation hinsichtlich der Übertragung in dem vorstehend bezeichneten zweiten Protokollstatusmodells Folgendes beinhaltet:

unter Verwendung von Statusfallenoperator und/oder Eingabeveränderungsoperator, Durchführen der Verhaltensmutationsoperation hinsichtlich der vorstehend bezeichneten Übertragung, wobei der vorstehend bezeichnete Statusfallenoperator verwendet wird, um den Status nach erfolgter Übertragung in einen nicht ordnungsgemäßen Status zu verändern, während der vorstehend bezeichnete Eingabeveränderungsoperator verwendet wird, um die Eingabemeldungen der Übertragung in andere Meldungen zu verändern.

[0014] Hierbei wird vorgesehen, dass die vorstehend bezeichnete Ausführung der Verhaltensmutationsoperation hinsichtlich der logischen Operationen bei der Übertragung in dem vorstehend bezeichneten zweiten Protokollstatusmodells Folgendes beinhaltet:

unter Verwendung von einem beliebigen der nachstehend aufgeführten Operatoren beziehungsweise einer beliebigen Kombination der nachstehend aufgeführten Operatoren: Expression Negation Operator, Expression Increase Operator, Expression Decrease Operator, stuck-at-0-Operator, stuck-at-1-Operator, Operand Negation Operator, Operand Reference Operator, Range Exploration Operator und Case Injection Operator, Ausführen der Verhaltensmutationsoperation hinsichtlich der vorstehend bezeichneten logischen Operationen, wobei:

der vorstehend bezeichnete Expression Negation Operator verwendet wird, um den Expressionswert in der logischen Operation umzukehren;

der vorstehend bezeichnete Expression Increase Operator verwendet wird, um den Expressionswert in der logischen Operation um n zu vermehren;

der vorstehend bezeichnete Expression Decrease Operator verwendet wird, um den Expressionswert in der logischen Operation um n zu vermindern;

der vorstehend bezeichnete stuck-at-0-Operator verwendet wird, um den Expressionswert in der logischen Operation auf 0 zu verändern,

der vorstehend bezeichnete stuck-at-1-Operator verwendet wird, um den Expressionswert in der logischen Operation auf 1 zu verändern;

der vorstehend bezeichnete Operand Negation Operator verwendet wird, um die verschiedenen Operanden in der Expression nach Stellen umzukehren;

der vorstehend bezeichnete Operand Reference Operator verwendet wird, um die Operanden nacheinander mit ihrem nachfolgenden Operanden auszutauschen und den letzten Operanden mit dem ersten Operanden auszutauschen;

der vorstehend bezeichnete Range Exploration Operator verwendet wird, um den Wert von jedem Operanden in der Expression nach Stellen umzukehren und die betreffende Zuweisungsanweisung mit der Kennzeichnung "Ersetzung" zu versehen;

und der vorstehend bezeichnete Case Injection Operator verwendet wird, um in der logischen Operation befindliche, in ihrer Länge veränderbare Felder um n Byte zu vermehren oder zu vermindern und die betreffende Zuweisungsanweisung mit der Kennzeichnung "Injektion" zu versehen, wobei es sich bei n um eine positive ganze Zahl handelt.

[0015] Hierbei wird vorgesehen, dass jedes der für den Modelltest erzeugten Gegenbeispiele zu einer Testabfolge abgebildet wird, wobei das betreffende Gegenbeispiel einem in dem vorstehend bezeichneten mutierten Protokollstatusmodell vorhandenen Übertragungsweg entspricht, welcher von einer Übertragung oder einer Mehrzahl von solchen Übertragungen gebildet wird, wobei auf dem vorstehend bezeichneten Übertragungsweg eine Eingabemeldung oder eine Mehrzahl von solchen Eingabemeldungen sowie eine diesen entsprechende Ausgabemeldung oder eine Mehrzahl von solchen Ausgabemeldungen existiert;

wobei das vorstehend bezeichnete Abbilden des vorstehend bezeichneten Gegenbeispiels zu einer Testabfolge Folgendes beinhaltet: Abbilden des vorstehend bezeichneten Gegenbeispiels zu einer auf dem vorstehend

bezeichneten Übertragungsweg existierender Eingabemeldung oder einer Mehrzahl von solchen Eingabemeldungen sowie einer diesen entsprechenden Ausgabemeldung oder einer Mehrzahl von solchen Ausgabemeldungen, wobei die Abfolge der erhaltenen Meldungen die Abfolge der vorstehend bezeichneten Testabfolge ist, und Kennzeichnen einer einer beliebigen, den kategorischen Festlegungen widersprechenden mutierten Übertragung entsprechenden Meldung sowie der in den vorstehend bezeichneten Meldungen durch Modelltest ermittelten Felder.

[0016] Hierbei wird vorgesehen, dass die vorstehend bezeichnete Kennzeichnung einer einer beliebigen, den kategorischen Festlegungen widersprechenden mutierten Übertragung entsprechenden Meldung sowie der in den vorstehend bezeichneten Meldungen durch Modelltest ermittelten Felder Folgendes beinhaltet:

falls der Modelltest die Identifizierung der Mutation eines gewissen Feldes in der vorstehend bezeichneten Meldung mit möglicher Herbeiführung des betreffenden Gegenbeispiels ermöglicht und das vorstehend bezeichnete Feld über keine andere Kennzeichnung verfügt, Kennzeichnen des vorstehend bezeichneten Feldes als "Fuzz-Testing", während anderenfalls die gesamte Meldung mit der Kennzeichnung "Fuzz-Testing" versehen wird;

falls der Modelltest die Identifizierung der Kennzeichnung einer einem gewissen Feld in der vorstehend bezeichneten Meldung entsprechenden logischen Operation als "Ersetzung" ergibt, Kennzeichnen für das vorstehend bezeichnete in der Meldung enthaltene Feld mit "Ersetzung"; und

falls der Modelltest die Identifizierung der Kennzeichnung einer einem gewissen Feld in der vorstehend bezeichneten Meldung entsprechenden logischen Operation als "Injektion" ergibt, Kennzeichnen für das vorstehend bezeichnete in der Meldung enthaltene Feld mit "Injektion".

[0017] Hierbei wird vorgesehen, dass bei der vorstehend bezeichneten Ausführung der Syntaxmutationsoperation hinsichtlich der vorstehend bezeichneten Testabfolge bezüglich von in der Testabfolge enthaltenen und mit der Kennzeichnung "Fuzz-Testing" versehenen Meldungen beziehungsweise bezüglich von mit der Kennzeichnung "Fuzz-Testing" versehene Felder enthaltenden Meldungen unter Verwendung des Syntaxmutationsoperators die Ausführung der Syntaxmutationsoperation erfolgt.

[0018] Hierbei wird vorgesehen, dass der vorstehend bezeichnete Syntaxmutationsoperator einen beliebige der nachstehend aufgeführten Syntaxmutationsoperatoren beziehungsweise eine beliebige Kombination der nachstehend aufgeführten Syntaxmutationsoperatoren beinhaltet:

Feldlöschungsoperator, welcher verwendet wird, um mit der Kennzeichnung "Fuzz-Testing" versehene Felder in den Meldungen zu löschen;

Feldwiederholungsoperator, welcher verwendet wird, um mit der Kennzeichnung "Fuzz-Testing" versehene Felder in den Meldungen zu mehreren Kopien zu kopieren;

Feldaustauschoperator, welcher verwendet wird, um die Position von mit der Kennzeichnung "Fuzz-Testing" versehenen Feldern mit der Position des folgenden Feldes auszutauschen;

Breitenveränderungsoperator, welcher verwendet wird, um den Inhalt von mit der Kennzeichnung "Fuzz-Testing" versehenen Feldern um n Byte zu vermehren beziehungsweise zu vermindern, wobei es sich bei n um eine positive ganze Zahl handelt;

Codeveränderungsoperator, welcher verwendet wird, um das Codeformat von mit der Kennzeichnung "Fuzz-Testing" versehenen Feldern zu einem anderen Format zu verändern; sowie

Wertnegierungsoperator, welcher verwendet wird, um den Wert von mit der Kennzeichnung "Fuzz-Testing" versehenen Feldern umzukehren beziehungsweise nach Stellen umzukehren.

[0019] Hierbei wird vorgesehen, dass bei der vorstehend bezeichneten Ausführung der Syntaxmutationsoperation hinsichtlich der vorstehend bezeichneten Testabfolge hinsichtlich von in Testanwendungsbeispielen vorhandenen, mit der Kennzeichnung "Ersetzung" versehenen Felder enthaltenden Meldungen unter Verwendung vorab definierter Testwerte beziehungsweise unter Verwendung von den kategorischen Festlegungen widersprechenden nicht ordnungsgemäßen Werten die Ersetzung der in der betreffenden Meldung mit der Kennzeichnung "Ersetzung" versehenen Felder erfolgt und die entsprechenden mutierten Meldungen erhalten werden;

wobei hinsichtlich von in Testanwendungsbeispielen vorhandenen, mit der Kennzeichnung "Injektion" versehenen Felder enthaltenden Meldungen unter Verwendung vorab definierter Testwerte beliebiger Länge beziehungsweise unter Verwendung von den kategorischen Festlegungen widersprechenden nicht ordnungsgemäßen Werten die Injektion in die in der betreffenden Meldung mit der Kennzeichnung "Injektion" versehenen Felder erfolgt und die entsprechenden mutierten Meldungen erhalten werden.

[0020] Hierbei wird vorgesehen, dass das betreffende Verfahren weiterhin Folgendes beinhaltet:

unmittelbares Verwenden von Testabfolgen, welche lediglich mit der Kennzeichnung "Ersetzung" versehene beziehungsweise mit der Kennzeichnung "Injektion" versehene Felder enthalten, als Testanwendungsbeispiele zur Durchführung von Fuzz-Testing.

[0021] Hierbei wird vorgesehen, dass die vorstehend bezeichnete Verwendung der vorstehend bezeichneten Testanwendungsbeispiele zur Durchführung von Fuzz-Testing hinsichtlich der Protokollimplementierung Folgendes beinhaltet:

Versenden der in den betreffenden Testanwendungsbeispielen enthaltenen jeweiligen Eingabemeldungen nacheinander an die vorstehend bezeichnete Protokollimplementierung; und hinsichtlich der von der vorstehend bezeichneten Protokollimplementierung empfangenen Reaktionsmeldungen gilt, dass wenn irgendeine der empfangenen Reaktionsmeldungen mit den in dem betreffenden Testanwendungsbeispiel enthaltenen Ausgabemeldungen nicht übereinstimmt, beziehungsweise sich die vorstehend bezeichnete Protokollimplementierung in einem nicht ordnungsgemäßen Status befindet, somit festgestellt wird, dass bei der vorstehend bezeichneten Protokollimplementierung den entsprechenden Eingabemeldungen entsprechende Sicherheitslücken existieren.

[0022] Hierbei wird vorgesehen, dass das betreffende Verfahren weiterhin beinhaltet, dass vor der Durchführung der Syntaxmutationsoperation eine Optimierungsbehandlung der vorstehend bezeichneten Testabfolge oder einer Mehrzahl von solchen Testabfolgen durchgeführt wird, um redundante Testabfolgen zu entfernen.

[0023] Hierbei wird vorgesehen, dass das vorstehend bezeichnete erste Protokollstatusmodell den betreffenden Protokollstatus und die Statusübertragung beschreibt.

[0024] Durch die praktischen Ausführungsbeispiele der vorliegenden Erfindung erfolgt außerdem die Bereitstellung einer Vorrichtung zur Durchführung von Sicherheitstests hinsichtlich der Protokollimplementierung, wobei vorgesehen wird, dass die betreffende Vorrichtung Folgendes beinhaltet:

ein erstes Modul (**601**), welches verwendet wird, um während der Durchführung des Sicherheitstests hinsichtlich der vorstehend bezeichneten Protokollimplementierung das Abbilden der vorstehend bezeichneten Protokollimplementierung zugrunde liegenden Protokollspezifikationen zu einem ersten Protokollstatusmodell vorzunehmen;

ein zweites Modul (**602**), welches verwendet wird, um in das von dem vorstehend bezeichneten ersten Modul (**601**) erhaltene vorstehend bezeichnete erste Protokollstatusmodell einen nicht ordnungsgemäßen Status einzuführen und das zweite Protokollstatusmodell zu erzeugen, wobei der vorstehend bezeichnete nicht ordnungsgemäße Status bei der vorstehend bezeichneten Protokollimplementierung existierenden Sicherheitsstörungen entspricht;

ein drittes Modul (**603**), welches verwendet wird, um unter Verwendung kategorischer Festlegungen die Repräsentierung von Testzielen des von dem vorstehend bezeichneten zweiten Modul (**602**) stammenden vorstehend bezeichneten zweiten Protokollstatusmodells und/oder des von dem vorstehend bezeichneten ersten Modul (**601**) stammenden vorstehend bezeichneten ersten Protokollstatusmodells vorzunehmen, wobei unter Verwendung syntaktischer Beschränkungen die Repräsentierung der syntaktischen Bestimmungen für ordnungsgemäße Eingabemeldungen des vorstehend bezeichneten zweiten Protokollstatusmodells und/oder des vorstehend bezeichneten ersten Protokollstatusmodells erfolgt;

ein viertes Modul (**604**), welches verwendet wird, um hinsichtlich des von dem vorstehend bezeichneten zweiten Modul (**602**) stammenden vorstehend bezeichneten zweiten Protokollstatusmodells die Verhaltensmutationsoperation vorzunehmen und auf diese Weise das mutierte Protokollstatusmodell zu erzeugen;

ein fünftes Modul (**605**), welches verwendet wird, um unter Verwendung der von dem vorstehend bezeichneten dritten Modul (**603**) erhaltenen kategorischen Festlegungen hinsichtlich des von dem vorstehend bezeichneten vierten Modul (**604**) erzeugten vorstehend bezeichneten mutierten Protokollstatusmodells den Modelltest durchzuführen und eine Testabfolge oder eine Mehrzahl von solchen Testabfolgen zu erzeugen, wobei die vorstehend bezeichnete Testabfolge oder eine Mehrzahl von solchen Testabfolgen durch Abbilden des in dem vorstehend bezeichneten mutierten Protokollstatusmodell enthaltenen, den vorstehend bezeichneten kategorischen Festlegungen widersprechenden Gegenbeispiels oder einer Mehrzahl von solchen Gegenbeispielen erhalten wird;

ein sechstes Modul (**606**), welches verwendet wird, um hinsichtlich der von dem vorstehend bezeichneten fünften Modul (**605**) stammenden Testabfolge die Syntaxmutationsoperation vorzunehmen und die mutierte Meldungen enthaltende mutierte Testabfolge zu erzeugen, wobei eine Testabfolge eine Mutationen enthaltende mutierte Testabfolge oder eine Mehrzahl von solchen Mutationen enthaltenden Testabfolgen erzeugt;

ein siebtes Modul (**607**), welches verwendet wird, um unter Verwendung der von dem vorstehend bezeichneten dritten Modul (**603**) erhaltenen syntaktischen Beschränkungen hinsichtlich der von dem vorstehend bezeichneten sechsten Modul (**606**) erzeugten mutierten Testabfolge den Syntaxtest durchzuführen und die in den

vorstehend bezeichneten mutierten Meldungen enthaltenen, den vorstehend bezeichneten syntaktischen Beschränkungen widersprechenden gültigen mutierten Meldungen zu erhalten, wobei die vorstehend bezeichnete gültige mutierte Meldungen enthaltenden mutierten Testabfolgen als Testanwendungsbeispiele verwendet werden; sowie

ein achttes Modul (**608**), welches verwendet wird, um unter Verwendung der von vorstehend bezeichnetem siebten Modul (**607**) erhaltenen Testanwendungsbeispiele hinsichtlich der Protokollimplementierung die Durchführung von Fuzz-Testing vorzunehmen, um zu testen, ob bei der vorstehend bezeichneten Protokollimplementierung einem Testanwendungsbeispiel oder einer Mehrzahl von solchen Testanwendungsbeispielen entsprechende Sicherheitslücken existieren.

[0025] Hierbei wird vorgesehen, dass das vorstehend bezeichnete dritte Modul (**603**) weiterhin verwendet wird, um die Optimierung der vorstehend bezeichneten kategorischen Festlegungen vorzunehmen und um redundante kategorische Festlegungen zu entfernen, beziehungsweise dass das vorstehend bezeichnete fünfte Modul (**605**) weiterhin verwendet wird, um die Optimierung des vorstehend bezeichneten Testanwendungsbeispiels oder einer Mehrzahl von solchen Testanwendungsbeispielen vorzunehmen, um redundante Testanwendungsbeispiele zu entfernen.

[0026] Hierbei wird vorgesehen, dass das vorstehend bezeichnete fünfte Modul (**605**) weiterhin verwendet wird, um unmittelbar die Verwendung von Testabfolgen, welche lediglich mit der Kennzeichnung "Ersetzung" versehene beziehungsweise mit der Kennzeichnung "Injektion" versehene Felder enthalten, als Testanwendungsbeispiele an das vorstehend bezeichnete achte Modul (**608**) zur Durchführung von Fuzz-Testing auszugeben.

[0027] Hierbei wird vorgesehen, dass das vorstehend bezeichnete erste Protokollstatusmodell den betreffenden Protokollstatus und die Statusübertragung beschreibt.

[0028] Durch die vorliegende Erfindung erfolgt außerdem die Bereitstellung eines computerlesbaren Mediums, in welchem computerlesbare Codes zur Ausführung des vorstehend bezeichneten Verfahrens gespeichert sind.

[0029] Durch die vorliegende Erfindung erfolgt außerdem die Bereitstellung eines Computerprogramms, welches computerlesbare Codes zur Ausführung des vorstehend bezeichneten Verfahrens beinhaltet.

[0030] Mittels durch die vorliegende Erfindung bereitgestellten Verfahrens und bereitgestellter Vorrichtung kann die Durchführung von Sicherheitstest hinsichtlich der Protokollimplementierung mit hoher Ausführungseffizienz bei geringen Kosten der praktischen Umsetzung durchgeführt werden. Hierbei ist der Quellcode des Computerprogramms der zu testenden Protokollimplementierung nicht erforderlich, um den Sicherheitstest der Protokollimplementierung zu verwirklichen.

Erläuterung der Abbildungen

[0031] Nachstehend aufgeführt erfolgt anhand der beigefügten Abbildungen eine detaillierte Beschreibung beispielhafter praktischer Ausführungsbeispiele der vorliegenden Erfindung, um einem durchschnittlichen Fachmann des betreffenden technischen Gebietes ein noch deutlicheres Verständnis der vorstehend bezeichneten und der weiteren Merkmale und Vorteile der vorliegenden Erfindung zu ermöglichen.

[0032] Bei [Abb. 1](#) handelt es sich um die Darstellung des Ablaufschemas des Verfahrens der praktischen Ausführungsbeispiele gemäß der vorliegenden Erfindung.

[0033] Bei [Abb. 2](#) handelt es sich um die Darstellung des Ablaufschemas eines konkreten praktischen Ausführungsbeispiels des Verfahrens gemäß der vorliegenden Erfindung.

[0034] Bei [Abb. 3](#) handelt es sich um die Darstellung des ursprünglichen Protokollstatusmodells des in [Abb. 2](#) gezeigten praktischen Ausführungsbeispiels.

[0035] Bei [Abb. 4](#) handelt es sich um die Darstellung nach der Einführung des nicht ordnungsgemäßen Status in das in [Abb. 3](#) gezeigte ursprüngliche Protokollstatusmodell.

[0036] Bei [Abb. 5](#) handelt es sich um die Darstellung des mutierten Protokollstatusmodells, welches nach der Durchführung der Mutationsoperation des in [Abb. 4](#) gezeigten ursprünglichen Protokollstatusmodells erhalten wurde.

[0037] Bei [Abb. 6](#) handelt es sich um die Darstellung der Konstruktion der Vorrichtung der praktischen Ausführungsbeispiele gemäß der vorliegenden Erfindung.

[0038] Hierbei bedeuten:

101 – Abbilden der Protokollspezifikationen, welche die Grundlage der vorstehend bezeichneten Protokollimplementierung darstellen, zu einem ersten Protokollstatusmodell.

102 – Einführung eines nicht ordnungsgemäßen Status in das betreffende Protokollstatusmodell.

103 – Repräsentierung der Testziele unter Verwendung von kategorischen Festlegungen, Repräsentierung der syntaktischen Bestimmungen unter Verwendung von syntaktischen Beschränkungen.

104 – Ausführung der Verhaltensmutationsoperation hinsichtlich des betreffenden Protokollstatusmodells mit eingeführtem nicht ordnungsgemäßen Status und Erzeugung eines mutierten Protokollstatusmodells.

105 – Durchführung von Modelltests hinsichtlich des mutierten Protokollstatusmodells und Erhalten einer Testabfolge oder einer Mehrzahl von solchen Testabfolgen

106 – Ausführung der Syntaxmutationsoperation hinsichtlich der Testabfolgen und Erzeugung mutierter Testabfolgen.

107 – Ausführung von Syntaxtests hinsichtlich der mutierten Testabfolgen und Erzeugung von den vorstehend bezeichneten syntaktischen Beschränkungen widersprechende mutierte Meldungen enthaltenden Testanwendungsbeispielen.

108 – Durchführung von Fuzz-Testing hinsichtlich der Protokollimplementierung unter Verwendung der Testanwendungsbeispiele, um zu testen, ob in der betreffenden Protokollimplementierung Sicherheitslücken existieren.

201 – Abbilden der Protokollspezifikationen, welche die Grundlage der im Test befindlichen Implementierung IUT darstellen, zu einem PEFSM.

202 – Einführung eines nicht ordnungsgemäßen Status in den betreffenden PEFSM.

203 – Repräsentierung der Testziele unter Verwendung von kategorischen Festlegungen, Repräsentierung der syntaktischen Bestimmungen unter Verwendung von syntaktischen Beschränkungen.

204 – Ausführung der Verhaltensmutationsoperation hinsichtlich des PEFSM mit eingeführtem nicht ordnungsgemäßen Status und Erzeugung eines mutierten PEFSM.

205 – Durchführung von Modelltests hinsichtlich des mutierten PEFSM und Erhalten einer Testabfolge oder einer Mehrzahl von solchen Testabfolgen.

206 – Ausführung der Syntaxmutationsoperation hinsichtlich der Testabfolgen und Erzeugung mutierter Testabfolgen.

207 – Durchführung von Syntaxtests hinsichtlich der mutierten Testabfolgen und Erzeugung von den vorstehend bezeichneten syntaktischen Beschränkungen widersprechende mutierte Meldungen enthaltenden Testanwendungsbeispielen.

208 – Durchführung von Fuzz-Testing hinsichtlich der im Test befindlichen Implementierung IUT unter Verwendung der Testanwendungsbeispiele, um zu testen, ob in der betreffenden im Test befindlichen Implementierung IUT Sicherheitslücken existieren.

601 – erstes Modul, **602** – zweites Modul, **603** – drittes Modul, **604** – viertes Modul, **605** – fünftes Modul, **606** – sechstes Modul, **607** – siebtes Modul, **608** – achtes Modul

Praktische Ausführungsbeispiele

[0039] Nachstehend aufgeführt erfolgt anhand der beigefügten Abbildungen und der praktischen Ausführungsbeispiele eine weitere Erläuterung der vorliegenden Erfindung. Es versteht sich, dass die vorliegend beschriebenen konkreten praktischen Ausführungsbeispiele lediglich der Erläuterung der vorliegenden Erfindung dienen, ohne eine Beschränkung der vorliegenden Erfindung darzustellen.

[0040] Durch die praktischen Ausführungsbeispiele der vorliegenden Erfindung erfolgt die Bereitstellung eines Verfahrens zur Durchführung von Sicherheitstests hinsichtlich der Protokollimplementierung (Protocol Implementation). Bei der Durchführung von Tests hinsichtlich der Protokollimplementierung beinhaltet das in [Abb. 1](#) gezeigte Ablaufschema des betreffenden Verfahrens die nachstehend aufgeführten Verfahrensschritte:

[0041] Verfahrensschritt **101**: Abbilden der Protokollspezifikationen, welche die Grundlage der vorstehend bezeichneten Protokollimplementierung darstellen, zu einem ersten Protokollstatusmodell (dies kann auch als ursprüngliches Protokollstatusmodell bezeichnet werden).

[0042] Verfahrensschritt **102**: Einführung eines nicht ordnungsgemäßen Status in das betreffende erste Protokollstatusmodell.

[0043] Verfahrensschritt **103**: Repräsentierung der Testziele von betreffendem zweiten Protokollstatusmodell und/oder betreffendem ersten Protokollstatusmodell unter Verwendung von kategorischen Festlegungen sowie gleichzeitig Repräsentierung der syntaktischen Bestimmungen für ordnungsgemäße Eingabemeldungen von betreffendem zweiten Protokollstatusmodell und/oder betreffendem ersten Protokollstatusmodell unter Verwendung von syntaktischen Beschränkungen.

[0044] Verfahrensschritt **104**: Durch Ausführung der Verhaltensmutationsoperation (Behavior Mutation Operation) hinsichtlich des betreffenden zweiten Protokollstatusmodells mit eingeführtem nicht ordnungsgemäßen Status erfolgt die Injektion von Sicherheitsstörungen in das betreffende Protokollstatusmodell und Erzeugung eines mutierten Protokollstatusmodells.

[0045] Verfahrensschritt **105**: Durchführung von Modelltests hinsichtlich des betreffenden mutierten Protokollstatusmodells und Erzeugung eines den vorstehend bezeichneten kategorischen Festlegungen widersprechenden Gegenbeispiels oder einer Mehrzahl von solchen Gegenbeispielen in dem betreffenden mutierten Protokollstatusmodell, wobei das vorstehend bezeichnete Gegenbeispiel oder eine Mehrzahl von solchen Gegenbeispielen zu einer Testabfolge oder einer Mehrzahl von solchen Testabfolgen abgebildet werden.

[0046] Verfahrensschritt **106**: Ausführung der Syntaxmutationsoperation (Syntax Mutation Operation) hinsichtlich der vorstehend bezeichneten Testabfolgen und Injektion syntaktischer Fehler in die in den betreffenden Testabfolgen enthaltenen Meldungen und Erzeugung mutierter Meldungen enthaltender mutierter Testabfolgen, wobei eine Testabfolge eine mutierte Meldungen enthaltende mutierte Testabfolge oder eine Mehrzahl von solchen mutierten Meldungen enthaltenden mutierten Testabfolgen erzeugen kann.

[0047] Verfahrensschritt **107**: Durchführung von Syntaxtests hinsichtlich in den vorstehend bezeichneten mutierten Testabfolgen enthaltenen mutierten Meldungen, um in den vorstehend bezeichneten mutierten Meldungen enthaltene und den vorstehend bezeichneten syntaktischen Beschränkungen widersprechende gültige mutierte Meldungen zu erhalten, wobei die vorstehend bezeichnete gültige mutierte Meldungen enthaltenden vorstehend bezeichneten mutierten Testabfolgen als Testanwendungsbeispiele verwendet werden.

[0048] Verfahrensschritt **108**: Durchführung von Fuzz-Testing hinsichtlich der betreffenden Protokollimplementierung unter Verwendung der vorstehend bezeichneten Testanwendungsbeispiele, um zu testen, ob in der betreffenden Protokollimplementierung Sicherheitslücken existieren.

[0049] Der Entwicklung von Protokollimplementierungen können verschiedene Protokollspezifikationen zugrunde liegen. Dies betrifft insbesondere Protokollimplementierungen für die Verbindung mehrerer verteilt angeordneter Computer sowie die betreffende Übertragung von Informationen und Daten. Die hierbei zu verwirklichenden Funktionen haben bestimmten standardisierten Bestimmungen des Protokolls zu entsprechen. Eine Protokollspezifikation beschreibt Syntax/Semantik für verschiedene Funktionsverhaltensweisen und Informationsübertragungen, welche bei der Protokollimplementierung verwirklicht werden können. Hierbei kann ein und dasselbe Funktionsverhalten konkret als verschiedene Protokollimplementierung mit verschiedenem Quellcode umgesetzt werden. Dies bedeutet mit anderen Worten, dass die Protokollspezifikationen keine Festlegung des Quellcodes der Protokollimplementierung bei der konkreten Umsetzung treffen und die Protokollspezifikationen deshalb als Referenz bei Black-box-Tests dienen.

[0050] Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung erfolgt in dem Bereich der Sicherheitstests der Protokollimplementierung die Verwendung eines auf Modellen beruhenden formalisierten Verfahrens, um Sicherheitslücken von auf verschiedenen Protokollen basierenden im Test befindlichen Implementierungen IUT zu identifizieren. Hierbei liegt die Beachtung auf Black-box-Tests und Syntax/Semantik (auch als Syntax bezeichnet) für durch Protokollspezifikationen beschriebene Funktionsverhaltensweisen und Informationsübertragung, ohne dass es erforderlich ist, den Quellcode oder die detaillierte Inhaltsstruktur der im Test befindlichen Implementierung IUT zu kennen, wobei gleichzeitig die Vorteile von Fuzz-Testing, Mutationsanalyse und Modelltests genutzt werden, um gegenseitig Mängel zu kompensieren und einen formalisierten Sicherheitstest des Protokolls der im Test befindlichen Implementierung IUT vorzunehmen. Das Verfahren ist einfach umzusetzen bei gleichzeitig geringen Kosten. Es ist außerdem eine hohe Ausführungseffizienz mit automatischem Test potentieller Sicherheitslücken in der im Test befindlichen Implementierung IUT gegeben.

[0051] Nachstehend aufgeführt erfolgt eine detaillierte Erläuterung der vorstehend bezeichneten verschiedenen Verfahrensschritte.

I. Verfahrensschritt 101

[0052] Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung kann es sich bei dem Protokollstatusmodell um einen endlichen Zustandsautomaten (FSM, Finite State Machine) (auch als endlicher Automat bezeichnet) handeln, beispielsweise um einen parameterisierten erweiterten endlichen Zustandsautomaten (PEFSM, Parameterized Extended Finite State Machine). Bei dem betreffenden Protokollstatusmodell kann es sich auch um eine andere Art von Modell handeln, beispielsweise um ein Input Output Labeled Transition System IOLTS (dessen Verwendung ebenfalls möglich ist). Die praktischen Ausführungsbeispiele der vorliegenden Erfindung stellen keinerlei Beschränkung hinsichtlich der konkreten Ausführung des Protokollstatusmodells dar. Jedes Modell, welches die Beschreibung von Protokollstatus und dessen Übertragung ermöglicht, fällt in den Bereich der vorstehend bezeichneten Protokollstatusmodelle.

[0053] Es wird angenommen, dass die Protokollspezifikationen zu einem PEFSM abgebildet werden. Der betreffende PEFSM enthält ein 6-Tupel (6-tuples), welches als $(S, s_0, \Sigma, \Gamma, \delta, \lambda)$ ausgedrückt werden kann. Hierbei repräsentiert S die Statuszusammenfassung des PEFSM (also die Zusammenfassung sämtlicher durch die Protokollspezifikationen festgelegter Zustände); s_0 repräsentiert den ursprünglichen Status des Protokolls; hinsichtlich sämtlicher Zustände repräsentiert der Vektor $\bar{v}$ die finite Zusammenfassung, einschließlich der internen Variablen mit ausgesparten Ursprungswerten. Σ repräsentiert die Zusammenfassung der Eingabemeldungen, wobei jede Eingabemeldung $i_k(\bar{v}_k)$ den Vektor (also den Parametervektor) $\bar{v}_k$ ihres Feldes enthält; Γ repräsentiert die Zusammenfassung der Ausgabemeldungen, wobei jede Ausgabemeldung $o_k(\bar{w}_k)$ den Vektor (also den Parametervektor) $\bar{w}_k$ ihres Feldes enthält; $\delta: S \times \Sigma \rightarrow S$ repräsentiert die Übertragungsfunktion, welche den zu übertragenden Zielstatus nach Erhalt einer Eingabemeldung $i_k(\bar{v}_k)$ ($\in \Sigma$) definiert; und $\lambda: S \times \Sigma \rightarrow \Gamma$ repräsentiert die Ausgabefunktion, welche die Ausgabemeldung $o_k(\bar{w}_k)$ ($\in \Gamma$) nach Erhalt einer Eingabemeldung $i_k(\bar{v}_k)$ definiert. In einem PEFSM umfasst einew Übertragung t : eine Gruppe der Überwachung unterliegender Bedingungen (können als "if statement" ausgedrückt werden) und eine Aktion $A(\bar{v}, i_k(\bar{v}_k), o_k(\bar{w}_k))$, wobei die betreffende Aktion eine Operation hinsichtlich Variable, Vektor und Eingabe/Ausgabe-Parameter darstellt. Wenn sämtliche der Überwachung unterliegende Bedingungen erfüllt sind, kann die betreffende Übertragung ausgeführt werden und die betreffende Aktion wird bereits basierend auf den aktuellen Werten von Variablen und Vektor sowie Eingabe/Ausgabe-Parametern ausgeführt. Somit wird die Übertragung des PEFSM von dem aktuellen Status in einen nächsten Status bewirkt.

[0054] Entsprechend den in den Protokollspezifikationen enthaltenen Definitionen können Festlegungen erfolgen hinsichtlich des Status, in welchem sich die im Test befindliche Implementierung IUT möglicherweise befindet, hinsichtlich der Übertragung zwischen verschiedenen Zuständen sowie hinsichtlich den der Übertragung entsprechenden Eingabemeldungen und Ausgabemeldungen, wobei anschließend entsprechend diesen festgelegten Zuständen, entsprechend der Übertragung und entsprechend den der Übertragung entsprechenden Eingabemeldungen und Ausgabemeldungen durch das vorstehend bezeichnete Verfahren die der im Test befindlichen Implementierung IUT zugrundeliegenden Protokollspezifikationen als entsprechende PEFSM formalisiert werden können.

[0055] Hierbei können die der Übertragung entsprechenden Eingabemeldungen und Ausgabemeldungen von der tatsächlichen Netzwerkmeldungsabfolge stammen. Beispielsweise können gegenwärtig vorhandene Werkzeuge zum Paket-Abgriff verwendet werden, etwa Werkzeuge zur Analyse von Netzwerk-Kommunikationsverbindungen wie Wireshark oder Tcpdump beziehungsweise andere entsprechende Werkzeuge.

[0056] Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung können Zuweisungsanweisungen beziehungsweise andere Formen verwendet werden, um die Darstellung der logischen Operationen in dem Protokollstatus vorzunehmen, einschließlich Eingabemeldungen (beziehungsweise darin enthaltene Felder) $i_k(\bar{v}_k)$, interne Variable $\bar{v}$ und Ausgabemeldungen (beziehungsweise darin enthaltene Felder), $o_k(\bar{w}_k)$, wobei das Format der Zuweisungsanweisungen normalerweise "Variable = Expression" ist.

II. Verfahrensschritt 102

[0057] Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung kann in das Protokollstatusmodell die Einführung von einem nicht ordnungsgemäßen Status oder einer Mehrzahl von solchen nicht ordnungsgemäßen Zuständen vorgenommen werden. Bei diesen nicht ordnungsgemäßen Zuständen kann es sich auch um als Störung bezeichnete Zustände beziehungsweise um ein sogenanntes "Wurmloch" (worm-

hole) handeln. Der betreffende nicht ordnungsgemäße Status entspricht einer bei der im Test befindlichen Implementierung IUT auftretenden Sicherheitsstörung (einschließlich Systemstörungen beziehungsweise in den Protokollspezifikationen nicht definierter Zustände, wie beispielsweise Absturz oder DoS usw.). Weil in den Protokollspezifikationen keine Definitionen von Sicherheitsstörungen, welche die im Test befindliche Implementierung IUT in einen Störungsstatus bringen, vorhanden sind, kann durch die Einführung eines nicht ordnungsgemäßen Status eine Sicherheitsstörungskomponente in das Protokollstatusmodell eingeführt werden, um den Sicherheitstest vorzunehmen.

[0058] Wenn hierbei die Einführung von einem einzelnen nicht ordnungsgemäßen Status erfolgt, betrifft dieser einzelne nicht ordnungsgemäße Status sämtliche Kategorien von Sicherheitsstörungen. Wenn die Einführung von mehreren nicht ordnungsgemäßen Zuständen erfolgt, betrifft jeder nicht ordnungsgemäße Status eine bestimmte Kategorie von Sicherheitsstörungen. In diesem Fall ist es erforderlich, über detaillierte Informationen bezüglich der Sicherheitsstörung dieser bestimmten Kategorie zu verfügen.

III. Verfahrensschritt 103

[0059] Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung können die bei der im Test befindlichen Implementierung IUT möglicherweise auftretenden Sicherheitsstörungen auch als abzudeckende Sicherheitsanforderungen bezeichnet werden. Die betreffenden Sicherheitsstörungen können als kategorische Festlegung (Predicate) der temporalen Logik definiert werden. Dies bedeutet, dass die kategorische Festlegung der temporalen Logik die Testziele des Protokollsicherheitsmodells repräsentiert. Hierbei kann es sich bei der temporalen Logik um eine lineare temporale (LTL, Linear Temporal Logic) oder um eine Rechenbaumlogik (CTL, Computation Tree Logic) usw. handeln. Die praktischen Ausführungsbeispiele der vorliegenden Erfindung stellen keine Beschränkung hinsichtlich der konkreten temporalen Logik dar.

[0060] Konkret wird vorgesehen, dass das Verfahren der Definition der Sicherheitsstörung als kategorische Festlegungen der temporalen Logik eine beliebige der nachstehend aufgeführten Positionen beziehungsweise eine beliebige Kombination der nachstehend aufgeführten Positionen darstellen kann, ohne allerdings hierauf beschränkt zu sein:

- 1) Definition der kategorischen Festlegungen basierend auf den gegebenen Testzielen;
- 2) Vorab-Definition der kategorischen Festlegungen gemäß aus dem technischen Bereich der Sicherheitstests stammendem Wissen;
- 3) Reflexion der kategorischen Festlegungen aus dem ursprünglichen Protokollstatusmodell (also aus dem durch Verfahrensschritt 101 erhaltenen ersten Protokollstatusmodell). Beispielsweise kann hinsichtlich jeder Übertragung in dem Protokollstatusmodell entsprechend der Erzeugung von kategorischen Festlegungen LTL beziehungsweise kategorischen Festlegungen CTL vorgenommen werden.

[0061] Bei der Durchführung der Sicherheitstests kann es sich bei den in der im Test befindlichen Implementierung IUT möglicherweise auftretenden Sicherheitsstörungen um eine Sicherheitsstörung oder um mehrere Sicherheitsstörungen handeln (normalerweise sind dies mehrere Sicherheitsstörungen). Auch bei den gemäß diesen Sicherheitsstörungen erhaltenen kategorischen Festlegungen kann es sich um eine kategorische Festlegung oder um mehrere kategorische Festlegungen handeln.

[0062] Ein Fachmann des betreffenden technischen Gebietes wird verstehen, dass es sich bei der kategorischen Festlegung um eine logische Beurteilung handelt, wobei eine bestimmte Prämisse als wahr beschrieben wird. Die kategorische Festlegung kommt durch die Verbindung einzelner oder kombinierter Behauptungen mittels Operatoren (beispielsweise und, oder, nicht) zustande, wobei das Ergebnis der Aufbereitung die zu beurteilenden Bedingungen als wahr (true) beziehungsweise als falsch (false) repräsentiert. Beispielsweise kann eine kategorische Festlegung eine einzelne Parameterfunktion sein, wobei diese Funktion über ein Shape Parameter, argument, verfügt. Wenn es sich bei dem betreffenden Parameter um eine positive Zahl handelt, ist der Umkehrwert der betreffenden Funktion wahr.

[0063] Ein Fachmann des betreffenden technischen Gebietes kann gemäß der vorstehend aufgeführten Beschreibung unter Nutzung des allgemeinen Fachwissens des betreffenden technischen Gebietes kategorische Festlegungen erhalten, welche die Repräsentierung der Testziele von betreffendem ersten Protokollstatusmodell und/oder betreffendem zweiten Protokollstatusmodell darstellen.

[0064] Gleichzeitig können hinsichtlich sämtlicher ordnungsgemäßer Eingabemeldungen entsprechend deren Protokollspezifikationen entsprechende syntaktische Beschränkungen definiert werden. Unter syntaktischen Beschränkungen werden syntaktische Bestimmungen verstanden, welche von ordnungsgemäßen Eingabe-

meldungen zu erfüllen sind. Konkret wird vorgesehen, dass das Verfahren der Definition syntaktischer Beschränkungen gemäß den Protokollspezifikationen eine beliebige der nachstehend aufgeführten Positionen eine beliebige Kombination der nachstehend aufgeführten Positionen beinhalten kann, ohne allerdings hierauf beschränkt zu sein:

- 1) Manuelle Definition syntaktischer Beschränkungen gemäß den Protokollspezifikationen, und
- 2) Automatische Reflexion der syntaktischen Beschränkungen aus den ordnungsgemäßen Ausgabemeldungen des ursprünglichen Protokollstatusmodells (also des gemäß Verfahrensschritt **101** erhaltenen ersten Protokollstatusmodells).

IV. Verfahrensschritt **104**

[0065] Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung können der vorstehend bezeichnete Verfahrensschritt **104** und der vorgelagerte Verfahrensschritt **103** gemeinsam ausgeführt werden. Es ist nicht erforderlich, die beiden Verfahrensschritte getrennt voneinander und nacheinander auszuführen.

[0066] Konkret wird vorgesehen, dass unter Verwendung eines Mutationsoperators (Mutation Operator) die Mutationsoperation hinsichtlich des Protokollstatusmodells durchgeführt wird. Hierbei kann der verwendete Mutationsoperator Folgendes umfassen, ohne allerdings hierauf beschränkt zu sein:

Statusfallenoperator (STO, State Trap Operator) und Eingabeveränderungsoperator (IAO, Input Alter Operator), Expressionsnegierungsoperator (ENO, Expression Negation Operator), Expressionsvermehrungsoperator (EIO, Expression Increase Operator), Expressionsverminderungsoperator (EDO, Expression Decrease Operator), 0-Beibehaltungsoperator (stuck-at-0), 1-Beibehaltungsoperator (stuck-at-1), Operandennegierungsoperator (ONO, Operand Negation Operator), Operandenreferenzoperator (ORO, Operand Reference Operator), Bereichsexplorationsoperator (REO, Range Exploration Operator) und Anwendungsbeispielinjektionsoperator (CIO, Case Injection Operator). Es ist möglich einen beliebigen der vorstehend aufgeführten Operatoren sowie eine beliebige Kombination der vorstehend aufgeführten Operatoren zu verwenden.

[0067] Hierbei umfasst der Mutationsoperator für die Ausführung der Mutation hinsichtlich der Übertragung, ohne allerdings hierauf beschränkt zu sein: Statusfallenoperator (STO) und Eingabeveränderungsoperator (IAO).

[0068] Der Statusfallenoperator (STO) wird verwendet, um den auf die Übertragung folgenden Status zu einem nicht ordnungsgemäßen Status zu verändern.

[0069] Der Eingabeveränderungsoperator (IAO) wird verwendet, um Eingabemeldungen der Übertragung in andere Eingabemeldungen zu verändern.

[0070] Bei logischen Operationen hinsichtlich der Übertragung, einschließlich Eingabemeldungen (beziehungsweise darin enthaltene Felder) $i_k(\bar{v}_k)$, interne Variable $\bar{v}$ und Ausgabemeldungen (beziehungsweise darin enthaltene Felder), $o_k(\bar{w}_k)$, wie beispielsweise Zuweisungsanweisungen, kann der ausgeführte Mutationsoperator einen beliebigen der nachstehend aufgeführten Operatoren beziehungsweise eine beliebige Kombination der nachstehend aufgeführten Operatoren umfassen, ohne allerdings hierauf beschränkt zu sein:

Expressionsnegierungsoperator (ENO), Expressionsvermehrungsoperator (EIO), Expressionsverminderungsoperator (EDO), 0-Beibehaltungsoperator (stuck-at-0), 1-Beibehaltungsoperator (stuck-at-1), Operandennegierungsoperator (ONO), Operandenreferenzoperator (ORO), Bereichsexplorationsoperator (REO) und Anwendungsbeispielinjektionsoperator (CIO).

[0071] Hierbei wird der Expressionsnegierungsoperator (ENO) verwendet, um den Expressionswert in der betreffenden Zuweisungsanweisung (rechts) (nach Stellen (Bit)) umzukehren.

[0072] Der Expressionsvermehrungsoperator (EIO) wird verwendet, um den Expressionswert in der betreffenden Zuweisungsanweisung (rechts) um n zu erhöhen, wobei es sich bei n um eine ganze positive Zahl handelt.

[0073] Der Expressionsverminderungsoperator (EDO) wird verwendet, um den Expressionswert in der betreffenden Zuweisungsanweisung (rechts) um n zu vermindern, wobei es sich bei n um eine ganze positive Zahl handelt.

[0074] Der 0-Beibehaltungsoperator (stuck-at-0) wird verwendet, um den Wert der Variablen in der betreffenden Zuweisungsanweisung (links) auf 0 zu verändern, wenn sämtlich 0 für die Variable (auf der linken Seite) zugewiesen wird.

[0075] Der 1-Beibehaltungsoperator (stuck-at-1) wird verwendet, um den Wert der Variablen in der betreffenden Zuweisungsanweisung (links) auf 1 zu verändern, wenn FF für die Variable (auf der linken Seite) zugewiesen wird.

[0076] Der Operandennegierungsoperator (ONO) wird verwendet, um die verschiedenen Operanden der Expression in der betreffenden Zuweisungsanweisung (rechts) nacheinander umzukehren beziehungsweise nach Stellen umzukehren.

[0077] Der Operandenreferenzoperator (ORO) wird verwendet, um die Operanden nacheinander mit ihrem nachfolgenden Operanden auszutauschen und den letzten Operanden mit dem ersten Operanden auszutauschen.

[0078] Der Bereichsexplorationsoperator (REO) wird verwendet, um den Wert von jedem Operanden der Expression in der betreffenden Zuweisungsanweisung nach Stellen umzukehren (also wie bei ENO) und die betreffende Zuweisungsanweisung mit der Kennzeichnung "fuzz:replace:n" zu versehen, wobei es sich bei n um die Bytebreite des Feldes handelt. Unter Feld wird hierbei die auf der linken Seite der Zuweisungsanweisung befindliche Variable verstanden. n beträgt normalerweise 1, 2, 4 oder 8 Byte. Hierbei ist "fuzz:replace:n" lediglich ein Anschauungsbeispiel. Bei der tatsächlichen praktischen Anwendung können auch andere Symbole zur Kennzeichnung verwendet werden. Es ist lediglich erforderlich, dass es sich um eine "Ersetzung" ausdrückende Kennzeichnung handelt.

[0079] Wenn angenommen wird, dass die Zuweisungsanweisung in der Länge variable Meldungsfelder enthält und dass eine der Länge dieser Felder entsprechende kategorische Festlegung existiert, so kann unter Verwendung des Anwendungsbeispielsinjektionsoperators (CIO) die Mutationsoperation hinsichtlich der betreffenden Zuweisungsanweisung ausgeführt werden. Der Anwendungsbeispielsinjektionsoperator (CIO) wird verwendet, um die in der Zuweisungsanweisung enthaltenen, in der Länge variablen Felder um n Byte zu vermehren und um die betreffende Zuweisungsanweisung mit der Kennzeichnung "fuzz:inject" zu versehen. Hierbei ist "fuzz:inject" lediglich ein Anschauungsbeispiel. Bei der tatsächlichen praktischen Anwendung können auch andere Symbole zur Kennzeichnung verwendet werden. Es ist lediglich erforderlich, dass es sich um eine "Injektion" ausdrückende Kennzeichnung handelt. Bei n handelt es sich hierbei um eine positive ganze Zahl.

[0080] Die praktischen Ausführungsbeispiele der vorliegenden Erfindung sind keineswegs auf die Verwendung konkreter Mutationsoperatoren beschränkt. Sämtliche für die Verwendung bei der Durchführung von Sicherheitstests geeignete Mutationsoperatoren fallen in den Bereich von Mutationsoperatoren gemäß den praktischen Ausführungsbeispielen der vorliegenden Erfindung.

[0081] Somit kann jedes Mal, wenn unter Verwendung eines Verhaltensmutationsoperators eine Verhaltensmutationsoperation ausgeführt wird, ein von dem ursprünglichen Protokollstatusmodell (also dem gemäß Verfahrensschritt 101 durch Abbilden erhaltenen Protokollstatusmodell) abweichendes mutiertes Protokollstatusmodell erzeugt werden. Diese mutierten Protokollstatusmodelle werden in den nachfolgenden Verfahrensschritten zur Ausführung der Modelltests verwendet.

V. Verfahrensschritt 105

[0082] Konkret wird vorgesehen, dass unter Verwendung der in Verfahrensschritt 103 erhaltenen kategorischen Festlegung der Modelltest von jedem in Verfahrensschritt 104 erhaltenen mutierten Protokollstatusmodell durchgeführt wird. Hierbei werden die vorstehend bezeichnete kategorische Festlegung und das mutierte Protokollstatusmodell als Eingabe für die Modellüberprüfungsvorrichtung verwendet. Die betreffende Modellüberprüfungsvorrichtung führt den Modelltest aus und erzeugt ein Gegenbeispiel (counterexample) oder eine Mehrzahl von solchen Gegenbeispielen, also Symbolabfolgen von Zuständen und Meldungen. Hierbei wird unter Gegenbeispiel ein vorstehend bezeichneten kategorischen Festlegungen widersprechendes Testanwendungsbeispiel verstanden, welches in Form von Übertragungswegen von der Modellüberprüfungsvorrichtung ausgegeben werden kann. Bei den sogenannten Übertragungswegen handelt es sich um durch eine Übertragung oder eine Mehrzahl von solchen Übertragungen von dem ursprünglichen Status des Protokollstatusmodells zu einem den kategorischen Festlegungen widersprechenden Status gebildete Wege. Entsprechend handelt es sich bei der letzten Übertragung in dem Übertragungsweg um die den kategorischen Festlegungen widersprechende mutierte Übertragung.

[0083] Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung wird vorgesehen, dass die verwendete Modellüberprüfungsvorrichtung (ohne allerdings hierauf beschränkt zu sein) auf Sprachen wie Neu-

Symbolmodellverifizierung (NuSMV, New Symbolic Model Verifier), Einfach-Promela- (eine Logiksprache) Interpreter (SPIN, Simple Promela Interpreter), Symbolanalyselabor (SAL, Symbolic Analysis Laboratory) basieren kann. Außerdem können die verwendeten Modelltestverfahren, ohne allerdings hierauf beschränkt zu sein, Folgendes umfassen: expliziten Modelltest (explicit model checking), symbolisierten Modelltest (symbolic model checking) oder begrenzten Modelltest (bounded model checking) usw. Wenn die Modellüberprüfungsvorrichtung vor der Eingabe des mutierten Protokollstatusmodells in die Modellüberprüfungsvorrichtung das Expressionsverfahren des betreffenden Protokollstatusmodells nicht identifizieren kann, kann unter Verwendung von Sprachen wie NuSMV oder Protokollmetasprache (PROMELA, Protocol Meta Language) die Übersetzung des betreffenden mutierten Protokollstatusmodells zu einem Kennzeichnungsübergangssystem (Labelled Transition System) erfolgen, um dieses der Modellüberprüfungsvorrichtung zur Bearbeitung zur Verfügung zu stellen.

[0084] Weil es sich bei dem durch den Modelltest erzeugten Gegenbeispiel oder einer Mehrzahl von solchen Gegenbeispielen lediglich um eine abstrakte Symbolabfolge handelt, ist es erforderlich, diese zu einer Testabfolge oder einer Mehrzahl von solchen Testabfolgen (testing trace) abzubilden, um anschließend Bearbeitung und Modelltest vorzunehmen. Bevorzugt wird vorgesehen, dass jedes Gegenbeispiel zu einer Testabfolge abgebildet wird. Konkret wird vorgesehen, dass, wie vorstehend aufgeführt, basierend auf einem mutierten Protokollstatusmodell ein Gegenbeispiel einem Übertragungsweg entspricht. Auf diesem Übertragungsweg existiert eine Reihe von Eingabemeldungen/Ausgabemeldungen, so dass die betreffenden Gegenbeispiele zu einer Testabfolge abgebildet werden können. In der betreffenden Testabfolge können die betreffenden Gegenbeispiele als eine Abfolge von Meldungen ausgedrückt werden, wobei die betreffende Abfolge von Meldungen Folgendes Folgendes beinhaltet: nach der Reihenfolge geordnete, sämtliche Eingabemeldungen und Ausgabemeldungen, welche auf dem Übertragungsweg, welcher den betreffenden Gegenbeispielen entspricht, existieren können. Dies bedeutet mit anderen Worten, dass die Testabfolge Folgendes beinhaltet: eine Reihe von Ausgabemeldungen zur Provokation von Eingabemeldungen der im Test befindlichen Implementierung IUT und entsprechende Ausgabemeldungen als erwartete Reaktion der im Test befindlichen Implementierung IUT.

[0085] Bei dem Vorgang des Abbildens der Gegenbeispiele zu Testabfolgen erfolgt die Kennzeichnung von jeder beliebigen Meldung, welche einer den kategorischen Festlegungen widersprechenden mutierten Übertragung entspricht, sowie der von der Modellüberprüfungsvorrichtung identifizierten Felder zwecks Verwendung bei den nachfolgenden Modelltests.

[0086] Das konkrete Kennzeichnungsverfahren umfasst, ohne allerdings hierauf beschränkt zu sein:

1. Falls die Modellüberprüfungsvorrichtung eine Identifizierung dahingehend vornimmt, dass die Mutation eines gewissen Feldes der betreffenden Meldung das betreffende Gegenbeispiel herbeiführen kann und das betreffende Feld über keine andere Kennzeichnung verfügt, erfolgt die Kennzeichnung des betreffenden Feldes mit "fuzz (Fuzz-Testing)"; anderenfalls erfolgt die Kennzeichnung der gesamten Meldung mit "fuzz". Hierbei ist "fuzz" lediglich ein Anschauungsbeispiel. Bei der tatsächlichen praktischen Anwendung können auch andere Symbole zur Kennzeichnung verwendet werden. Es ist lediglich erforderlich, dass es sich um eine die Erfordernis der Behandlung durch Fuzz-Testing ausdrückende Kennzeichnung handelt. In dem vorliegenden Text erfolgt die einfache Kennzeichnung mit "Fuzz-Testing".
2. Falls die Modellüberprüfungsvorrichtung eine Identifizierung der Kennzeichnungen von Feldern entsprechenden Zuweisungsanweisungen mit "fuzz:replace:n" vornimmt, erfolgt die Kennzeichnung des betreffenden Feldes in der Meldung mit "replace" und es erfolgt die Durchführung von Fuzz-Testing für das betreffende Feld unter Verwendung eines Testwertes mit Länge n beziehungsweise mit einem nicht ordnungsgemäßen Wert.
3. Falls die Modellüberprüfungsvorrichtung eine Identifizierung der Kennzeichnungen von Feldern entsprechenden Zuweisungsanweisungen mit "fuzz:inject" vornimmt, erfolgt die Kennzeichnung des betreffenden Feldes in der Meldung mit "inject" und es erfolgt die Durchführung von Fuzz-Testing für das betreffende Feld unter Verwendung sämtlicher Testwerte beziehungsweise nicht ordnungsgemäßer Werte.

[0087] Die vorstehend aufgeführten Symbolketten haben sämtlich beispielhaften Charakter.

[0088] Weiterhin wird vorgesehen, dass auch der Wertebereich der erhaltenen Felder der Meldungen, ordnungsgemäße Werte und kollidierende Werte das Datenaufkommen zu testender Fehlerwerte bei der Durchführung von Fuzz-Testing vermindern können. Dies bedeutet, dass hinsichtlich der identifizierten mit Kennzeichnung zu versehenen Felder von Meldungen eine Optimierungsbehandlung durchgeführt wird, um redundante Felder von Meldungen zu entfernen. Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung wird vorgesehen, dass sämtliche Verfahren gemäß dem gegenwärtig vorhandenen Stand der Technik für die Durchführung dieser Optimierungsbehandlung verwendet werden können, was an dieser Stelle keiner weiteren Erläuterung bedarf.

[0089] Weiterhin wird vorgesehen, dass zwecks Steigerung der Testeffizienz eine Optimierung der in Verfahrensschritt **104** erzeugten Testabfolge durchgeführt werden kann, um redundante Testabfolgen zu entfernen (also: um redundante Gegenbeispiele zu entfernen). Beispielsweise wird vorgesehen, dass vor der Durchführung des Modelltests eine Vereinfachungsbehandlung der kategorischen Festlegungen vorgenommen wird. Hierbei kann der Wirkungsbereich der kategorischen Festlegungen sowie deren Gehalt bezüglich Sicherheit/Logik analysiert werden. Entsprechend dem Ergebnis der Analyse können durch andere kategorische Festlegungen abgedeckte, also redundante kategorische Festlegungen gelöscht werden. Beziehungsweise ist es möglich, dass vor der Durchführung des Modelltests die Löschung redundanter Testabfolgen vorgenommen wird. Wenn es sich beispielsweise bei einem gewissen kurzen Gegenbeispiel (also einem Testweg) um das Präfix eines langen Gegenbeispiels handelt, so ist das betreffende kurze Gegenbeispiel redundant und die durch das betreffende kurze Gegenbeispiel abgebildete Testabfolge ist zu löschen. Diesbezüglich können bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung sämtliche Verfahren gemäß dem gegenwärtig vorhandenen Stand der Technik verwendet werden, um die Optimierungsbehandlung der Testabfolgen vorzunehmen, was an dieser Stelle keiner weiteren Erläuterung bedarf.

VI. Verfahrensschritt **106**

[0090] Hinsichtlich der in Verfahrensschritt **105** erzeugten Testanwendungsbeispiele erfolgt die Durchführung von Syntaxmutationen, um jeder Testabfolge entsprechende mutierte Testabfolgen zu erzeugen. Hierbei entspricht eine Testabfolge einer Testabfolge oder einer Mehrzahl von solchen Testabfolgen.

[0091] Konkret wird vorgesehen, dass mehrere Verfahren der Durchführung der Syntaxmutation hinsichtlich der Testabfolgen möglich sind, wovon nachstehend zwei Verfahren aufgeführt werden:

Erstes Verfahren:

[0092] Lediglich hinsichtlich von mit der Kennzeichnung "fuzz" versehenen Meldungen in jeder durch den Modelltest erzeugten Testabfolge beziehungsweise hinsichtlich von mit der Kennzeichnung "fuzz" versehene Felder enthaltenden Meldungen erfolgt die Durchführung der Syntaxmutation. Konkret wird vorgesehen (ohne allerdings hierauf beschränkt zu sein), dass ein beliebiger der nachstehend aufgeführten Syntaxmutationsoperatoren beziehungsweise eine beliebige Kombination der nachstehend aufgeführten Syntaxmutationsoperatoren verwendet wird, um die Syntaxmutationsoperation hinsichtlich der vorstehend bezeichneten Meldungen vorzunehmen und mutierte Meldungen zu erzeugen: Feldlöschungsoperator (Field Erase Operator), Feldwiederholungsoperator (Field Repeat Operator), Feldaustauschoperator (Field Exchange Operator), Breitenveränderungsoperator (Width Change Operator), Codeveränderungsoperator (Encode Change Operator) und Wertnegierungsoperator (Value Negation Operator).

[0093] Hierbei wird der Feldlöschungsoperator (Field Erase Operator) verwendet, um entsprechende mit der Kennzeichnung "fuzz" versehene Felder in den Meldungen zu löschen.

[0094] Der Feldwiederholungsoperator (Field Repeat Operator) wird verwendet, um mit der Kennzeichnung "fuzz" versehene Felder in den Meldungen zu n Kopien zu kopieren, wobei es sich bei n um eine positive ganze Zahl handelt.

[0095] Der Feldaustauschoperator (Field Exchange Operator) wird verwendet, um die Position von aktuell mit der Kennzeichnung "fuzz" versehenen Feldern mit der Position des nachfolgenden Feldes auszutauschen.

[0096] Der Breitenveränderungsoperator (Width Change Operator) wird verwendet, um den Inhalt von aktuell mit der Kennzeichnung "fuzz" versehenen Feldern um n Byte zu vermehren, wobei es sich bei n nicht lediglich um eine positive ganze Zahl handelt, sondern auch um eine negative ganze Zahl handeln kann.

[0097] Der Codeveränderungsoperator (Encode Change Operator) wird verwendet, um das Codeformat von mit der Kennzeichnung "fuzz" versehenen Feldern zu einem anderen Format zu verändern, beispielsweise von ASCII zur UTF-8, UNICODE usw.

[0098] Der Wertnegierungsoperator (Value Negation Operator) wird verwendet, um den Wert von mit der aktuellen Kennzeichnung "fuzz" versehenen Feldern umzukehren beziehungsweise nach Stellen umzukehren.

[0099] Auf diese Weise kann jedes Mal, wenn unter Verwendung eines Syntaxmutationsoperators eine Syntaxmutationsoperation ausgeführt wird, eine mutierte Meldung erzeugt werden, wobei gleichzeitig eine von der

ursprünglichen Testabfolge (also der gemäß Verfahrensschritt **105** durch Abbilden von jedem Gegenbeispiel erhaltenen Testabfolge) abweichende mutierte Testabfolge erhalten wird. Diese mutierten Testabfolgen werden in den nachfolgenden Verfahrensschritten zur Ausführung der Syntaxtests verwendet.

[0100] In diesem Falle ist es hinsichtlich von durch den Modelltest erzeugten Testabfolgen, welche lediglich die Kennzeichnung "replace (Ersetzung)" beziehungsweise "inject (Injektion)" aufweisende Felder enthalten, aber keine die Kennzeichnung "fuzz" aufweisende Felder enthalten, nicht erforderlich, den vorliegenden Verfahrensschritt auszuführen, sondern es erfolgt unmittelbar die Verwendung als Testanwendungsbeispiel für die Durchführung von Fuzz-Testing gemäß Verfahrensschritt **108**.

Zweites Verfahren:

[0101] Die Verwendung eines beliebigen der vorstehend aufgeführten Syntaxmutationsoperatoren beziehungsweise einer beliebigen Kombination der nachstehend aufgeführten Syntaxmutationsoperatoren erfolgt für die Durchführung der Syntaxmutationsoperation nicht nur hinsichtlich von jeder in der durch den Modelltest erzeugten Testabfolge mit der Kennzeichnung "fuzz" versehenen Meldung beziehungsweise von die Kennzeichnung "fuzz" enthaltenden Feldern, sondern auch hinsichtlich von jedem in der durch den Modelltest erzeugten Testabfolge enthaltenen Feld mit der Kennzeichnung "replace". Falls die Länge des betreffenden Feldes n beträgt, wird die Länge n als vorab definierter Testwert beziehungsweise als der kategorischen Festlegung widersprechender, nicht ordnungsgemäßer Wert zur Ersetzung von dem in der betreffenden Eingabemeldung enthaltenen mit der Kennzeichnung "replace" versehenen Feld vorgesehen, um mutierte Meldungen zur Verwendung bei dem Fuzz-Testing zu erzeugen. Hinsichtlich von jedem in der durch den Modelltest erzeugten Testabfolge mit der Kennzeichnung "inject" versehenen Feld kann ein unter Verwendung einer beliebigen Länge vorab definierter Wert beziehungsweise ein der kategorischen Festlegung widersprechender nicht ordnungsgemäßer Wert in die in den Ausgabemeldungen enthaltenen mit der Kennzeichnung "inject" versehenen Felder injiziert werden, um mutierte Meldungen zur Verwendung bei dem Fuzz-Testing zu erzeugen.

VII. Verfahrensschritt **107**

[0102] Hinsichtlich der in den vorstehend bezeichneten erzeugten mutierten Testabfolgen enthaltenen mutierten Meldungen erfolgt unter Verwendung der in Verfahrensschritt **103** reflektierten syntaktischen Beschränkungen die Durchführung des Syntaxtests. Falls die mutierten Meldungen zumindest einer syntaktischen Beschränkung widersprechen, gilt die betreffende mutierte Meldung als gültige mutierte Meldung. Gleichzeitig werden die betreffende gültige mutierte Meldungen enthaltenden Testabfolgen als Testanwendungsbeispiele bei dem nachfolgenden Fuzz-Testing verwendet.

VIII. Verfahrensschritt **108**

[0103] Konkret wird vorgesehen, dass hinsichtlich des ersten Falles von Verfahrensschritt **106** unter Verwendung von Testanwendungsbeispielen hinsichtlich der im Test befindlichen Implementierung IUT die Durchführung von Fuzz-Testing erfolgt, wobei das entsprechende Verfahren Folgendes umfasst:

- 1) Hinsichtlich von in den Eingabemeldungen enthaltenen mit der Kennzeichnung "replace" beziehungsweise "inject" versehenen Feldern erfolgt Ersetzung durch den vorab definierten Testwert beziehungsweise den der kategorischen Festlegung widersprechenden nicht ordnungsgemäßen Wert oder Injektion in das Feld der betreffenden Eingabemeldung, um eine mutierte Meldung zur Verwendung bei dem Fuzz-Testing zu erzeugen. Hierbei ist es hinsichtlich von in den Eingabemeldungen enthaltenen mit der Kennzeichnung "replace" versehenen Feldern erforderlich, unter Verwendung des Testwerts beziehungsweise des der kategorischen Festlegung widersprechenden nicht ordnungsgemäßen Wertes die Ersetzung des Feldes der betreffenden Eingabemeldung vorzunehmen, um eine mutierte Meldung zur Verwendung bei dem Fuzz-Testing zu erzeugen. Hinsichtlich von in der Eingabemeldung enthaltenen mit der Kennzeichnung "inject" versehenen Feldern kann unter Verwendung eines Testwertes beliebiger Länge beziehungsweise eines der kategorischen Festlegung widersprechenden nicht ordnungsgemäßen Wertes die Injektion in die in der betreffenden Eingabemeldung enthaltenen mit der Kennzeichnung "inject" versehenen Felder erfolgen, um eine mutierte Meldung zur Verwendung bei dem Fuzz-Testing zu erzeugen.
- 2) An die im Test befindliche Implementierung IUT erfolgt nacheinander die Versendung von einer in dem Testanwendungsbeispiel enthaltenen Testmeldung oder einer Mehrzahl von solchen Testmeldungen. Hierbei werden unter den sogenannten Testmeldungen den vorstehend bezeichneten Testanwendungsbeispielen entsprechende Eingabemeldungen verstanden; und
- 3) Hinsichtlich der von der im Test befindlichen Implementierung IUT empfangenen Reaktionsmeldungen wird Folgendes vorgesehen: hinsichtlich jeder empfangenen Reaktionsmeldung erfolgt ein Test, ob die be-

treffende Reaktionsmeldung mit den in den Testanwendungsbeispielen enthaltenen erwarteten Ausgabemeldungen übereinstimmt. Falls die getesteten Reaktionsmeldungen nicht mit den erwarteten Ausgabemeldungen übereinstimmen, beziehungsweise sich die im Test befindliche Implementierung IUT in einem nicht ordnungsgemäßen Status befindet, kann eine Beurteilung dahingehend vorgenommen werden, dass bei der im Test befindlichen Implementierung IUT möglicherweise eine entsprechende Sicherheitslücke existiert.

[0104] Hierbei wird vorgesehen, dass es sich bei dem der kategorischen Festlegung widersprechenden nicht ordnungsgemäßen Wert um einen außerhalb des Bereiches der ordnungsgemäßen Werte ausgewählten Wert beziehungsweise um einen kollidierenden Wert beziehungsweise um einen zufällig erzeugten Wert oder um einen vorab definierten Wert beziehungsweise um einen auf andere Weise erzeugten Wert handelt. Außerdem können der kategorischen Festlegung widersprechende Werte manchmal durch Gegenbeispiele und/oder mutierten PEFSM ausgegeben werden.

[0105] Konkret wird vorgesehen, dass hinsichtlich des zweiten Falles von Verfahrensschritt **106** unter Verwendung von Testanwendungsbeispielen hinsichtlich der im Test befindlichen Implementierung IUT die Durchführung von Fuzz-Testing erfolgt, wobei das entsprechende Verfahren Folgendes umfasst:

- 1) An die im Test befindliche Implementierung IUT erfolgt nacheinander die Versendung einer in dem Testanwendungsbeispiel enthaltenen Testmeldung oder einer Mehrzahl von solchen Testmeldungen. Hierbei werden unter den sogenannten Testmeldungen den vorstehend bezeichneten Testanwendungsbeispielen entsprechende Eingabemeldungen verstanden; und
- 2) Hinsichtlich der von der im Test befindlichen Implementierung IUT empfangenen Reaktionsmeldungen wird Folgendes vorgesehen: hinsichtlich jeder empfangenen Reaktionsmeldung erfolgt ein Test, ob die betreffende Reaktionsmeldung mit den in den Testanwendungsbeispielen enthaltenen erwarteten Ausgabemeldungen übereinstimmt. Falls die getesteten Reaktionsmeldungen nicht mit den erwarteten Ausgabemeldungen übereinstimmen, beziehungsweise sich die im Test befindliche Implementierung IUT in einem nicht ordnungsgemäßen Status befindet, kann eine Beurteilung dahingehend vorgenommen werden, dass bei der im Test befindlichen Implementierung IUT möglicherweise eine entsprechende Sicherheitslücke existiert.

[0106] Nach der Versendung der in den vorstehend bezeichneten Testanwendungsbeispielen enthaltenen mutierten Meldungen an die im Test befindliche Implementierung IUT kann durch Versendungsexplorationsmeldungen oder manuellen Test der Status der im Test befindlichen Implementierung IUT geprüft werden. Falls sich die im Test befindliche Implementierung IUT in nicht ordnungsgemäßem Status befindet, beispielsweise bei Systemstörungen, unbekanntem Status, Absturz, Programmversagen beziehungsweise keiner weiteren Reaktion auf Eingabemeldungen usw. wird eine Sicherheitslücke festgestellt und das aktuelle Testanwendungsbeispiel kann als Missbrauchsscript (misusing script) für die Feststellung von Sicherheitslücken dienen.

[0107] Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung wird vorgesehen, dass bei den vorstehend bezeichneten Verfahrensschritten **104** bis **108** die beiden folgenden Modi für die Ausführung verwendet werden können:

1. Gruppenbehandlungsmodus (Batch Mode). Dies bedeutet, dass die erste Gruppe sämtliche Testanwendungsbeispiele erzeugt, um anschließend die Durchführung von Fuzz-Testing vorzunehmen.

[0108] Konkret werden verschiedene Arten des Gruppenbehandlungsmodus vorgesehen, wovon nachstehend zwei Arten beispielhaft aufgeführt werden:

- 1) Zunächst erfolgt die Wiederholung der Verfahrensschritte **104** bis **105**, bis sämtliche Testanwendungsbeispiele für das mutierte Protokollstatusmodell erzeugt worden sind. Anschließend erfolgt die Durchführung der Verfahrensschritte **106** bis **107**, bis die Testanwendungsbeispiele für sämtliche Testabfolgen erzeugt worden sind. Anschließend erfolgt die Durchführung von Verfahrensschritt **108**, um hinsichtlich sämtlicher erzeugter Testanwendungsbeispiele nacheinander die Durchführung von Fuzz-Testing vorzunehmen.

[0109] Hierbei erfolgt unter Verwendung eines Verhaltensmutationsoperators die Durchführung der Verhaltensmutationsoperation hinsichtlich einer in dem Protokollstatusmodell enthaltenen Übertragung zur Erzeugung eines mutierten Protokollstatusmodells. Wenn sämtliche aktuell zu verwendenden Kombinationen von Verhaltensmutationsoperatoren und Übertragungen bereits für die Erzeugung von mutierten Protokollstatusmodellen verwendet worden sind, kann die Feststellung erfolgen, dass bereits sämtliche mutierte Protokollstatusmodelle erzeugt worden sind. Außerdem erfolgt unter Verwendung eines Syntaxmutationsoperators die Durchführung der Syntaxmutationsoperation hinsichtlich von in jeder der durch den Modelltest erhaltenen Testabfolgen mit Kennzeichnung versehenen Meldungen zur Erzeugung einer mutierten Meldung. Gleichzeitig wird

eine die betreffende mutierte Meldung enthaltende mutierte Testabfolge erhalten. Wenn sämtliche aktuell zu verwendenden Kombinationen von Syntaxmutationsoperatoren und Testabfolgen bereits für die Erzeugung von mutierten Testabfolgen verwendet worden sind, kann die Feststellung erfolgen, dass bereits sämtliche mutierte Testabfolgen erzeugt worden sind.

2) Es erfolgt die Wiederholung der Verfahrensschritte **104** bis **107**, bis sämtliche Testabfolgen für das mutierte Protokollstatusmodell erzeugt worden sind. Anschließend erfolgt die Durchführung von Verfahrensschritt **108**, um hinsichtlich sämtlicher erzeugter Testanwendungsbeispiele nacheinander die Durchführung von Fuzz-Testing vorzunehmen. Hierbei können die Verfahrensschritte **104–105** und die Verfahrensschritte **106–107** interaktiv ausgeführt werden.

[0110] Dies bedeutet, dass jedes Mal, wenn bei Verfahrensschritt **104** durch Verhaltensmutationsoperation ein mutiertes Protokollstatusmodell erhalten wird, in Verfahrensschritt **105** hinsichtlich dieses mutierten Protokollstatusmodells die Durchführung des Modelltests erfolgt, um ein Gegenbeispiel oder eine Mehrzahl von solchen Gegenbeispielen zu erzeugen. In Verfahrensschritt **106** wird durch Ausführung der Syntaxmutationsoperation die jedem Gegenbeispiel entsprechende mutierte Testabfolge erhalten. In Verfahrensschritt **107** erfolgt die Durchführung des Syntaxtests hinsichtlich der vorstehend bezeichneten mutierten Testabfolgen. Nach der Festlegung sämtlicher Testanwendungsbeispiele erfolgt Rückkehr zu Verfahrensschritt **104**, bis sämtliche Testanwendungsbeispiele der Testabfolge hinsichtlich des mutierten Protokollstatusmodells erzeugt worden sind. Anschließend erfolgt die Durchführung von Verfahrensschritt **108**, um hinsichtlich sämtlicher erzeugter Testanwendungsbeispiele nacheinander die Durchführung von Fuzz-Testing vorzunehmen.

2. Interaktivmodus (Interactive mode)

[0111] Dies bedeutet, dass die Erzeugung der Testanwendungsbeispiele und das Fuzz-Testing verzahnt durchgeführt werden. Konkret werden verschiedene Arten des Interaktivmodus vorgesehen, wovon nachstehend zwei Arten beispielhaft aufgeführt werden:

1) Es wird vorgesehen, dass bei jedem Verfahrensschritt **104** durch Verhaltensmutationsoperation ein mutiertes Protokollstatusmodell erhalten wird und dass in Verfahrensschritt **105** hinsichtlich dieses mutierten Protokollstatusmodells die Durchführung des Modelltests erfolgt, um ein Gegenbeispiel oder eine Mehrzahl von solchen Gegenbeispielen zu erzeugen. In Verfahrensschritt **106** wird durch Ausführung der Syntaxmutationsoperation die jedem Gegenbeispiel entsprechende mutierte Testabfolge oder eine Mehrzahl von solchen mutierten Testabfolgen erhalten. In Verfahrensschritt **107** erfolgt die Durchführung des Syntaxtests hinsichtlich der vorstehend bezeichneten Testabfolge oder einer Mehrzahl von solchen mutierten Testabfolgen. Bei der Festlegung eines Testanwendungsbeispiels erfolgt in Verfahrensschritt **108** die Verwendung des betreffenden Testanwendungsbeispiels zur Durchführung von Fuzz-Testing. Nach der Durchführung von Fuzz-Testing hinsichtlich sämtlicher Testanwendungsbeispiele von jedem Gegenbeispiel aus Verfahrensschritt **105** erfolgt Rückkehr zu Verfahrensschritt **104** mit Festlegung einer neuen Verhaltensmutationsoperation zur Festlegung des nächsten mutierten Protokollstatusmodells. Es erfolgt die Wiederholung der vorstehend bezeichneten Operation. Nach Durchführung von Fuzz-Testing hinsichtlich sämtlicher Gegenbeispiele sämtlicher Protokollstatusmodellen entsprechender Testanwendungsbeispiele erfolgt die Beendigung.

2) Es wird vorgesehen, dass bei jedem Verfahrensschritt **104** durch Verhaltensmutationsoperation ein mutiertes Protokollstatusmodell erhalten wird und dass in Verfahrensschritt **105** hinsichtlich dieses mutierten Protokollstatusmodells die Durchführung des Modelltests erfolgt, um ein Gegenbeispiel zu erzeugen. Weiterhin erfolgt in Verfahrensschritt **106** durch Ausführung der Syntaxmutationsoperation das Erhalten einer mutierten Testabfolge. Bei der Festlegung der betreffenden mutierten Testabfolge als Testanwendungsbeispiel durch Verfahrensschritt **107** erfolgt in Verfahrensschritt **108** die Verwendung des betreffenden Testanwendungsbeispiels zur Durchführung von Fuzz-Testing. Deshalb verhält es sich so, dass nach der erfolgten Durchführung von Fuzz-Testing unter Verwendung des aktuellen Testanwendungsbeispiels die Rückkehr zu Verfahrensschritt **106** zur Erzeugung der nächsten mutierten Testabfolge erfolgt. In Verfahrensschritt **107** erfolgt die Durchführung des Syntaxtests hinsichtlich der vorstehend bezeichneten mutierten Testabfolge. Bei der Festlegung der betreffenden mutierten Testabfolge als Testanwendungsbeispiel erfolgt in Verfahrensschritt **108** die Durchführung von Fuzz-Testing. Dieser Vorgang wird wiederholt, bis die Erzeugung sämtlicher möglicher Testanwendungsbeispiele sämtlicher Testabfolgen aus Verfahrensschritt **105** erfolgt ist sowie deren Test durchgeführt worden ist. Es erfolgt Rückkehr zu Verfahrensschritt **104** zur Erzeugung des nächsten mutierten Protokollstatusmodells und zur Wiederholung des vorstehend bezeichneten Vorgangs, bis die Erzeugung sämtlicher Testanwendungsbeispiele sämtlicher Testabfolgen der vorstehend bezeichneten mutierten Protokollstatusmodelle erfolgt ist und der Test beendet ist.

[0112] Ein Fachmann des betreffenden technischen Gebietes wird verstehen, dass es sich bei dem Protokollsicherheitstest darum handelt, dass an die im Test befindliche Implementierung IUT die Eingabe von in

den Protokollspezifikationen nicht definierten Eingabemeldungen erfolgt, was für die im Test befindliche Implementierung IUT ein unvorhersehbares Verhalten darstellt. Andererseits erfolgt ein Konsistenztest der im Test befindlichen Implementierung IUT, um festzulegen, ob die im Test befindliche Implementierung IUT der in ihrer Protokollspezifikation festgelegten technischen Konzeption entspricht. Die hauptsächliche Zielsetzung des Konsistenztests besteht darin, nachzuweisen, ob die in den Protokollspezifikationen festgelegten Funktionen/Verhaltensweisen von der im Test befindlichen Implementierung IUT bereits korrekt verwirklicht werden. Somit hat die im Test befindliche Implementierung IUT bei der Durchführung von Fuzz-Testing während der Ausgabe von in der Protokollspezifikation definierten Eingabemeldungen die in der Protokollspezifikation definierten korrekten Ausgabemeldungen zurückzumelden. Daher liegt bei dem Konsistenztest das Augenmerk auf den in den Protokollspezifikationen definierten Eingabemeldungen.

[0113] Bei dem vorstehend aufgeführten praktischen Ausführungsbeispiel gemäß der vorliegenden Erfindung werden die Protokollspezifikationen zu einem Protokollstatusmodell formalisiert. Dies kann als formalisierte Spezifikation S bezeichnet werden. Auf der Basis der betreffenden formalisierten Spezifikation S kann die automatische Erzeugung von Testanwendungsbeispielen für das Fuzz-Testing erfolgen, wobei die betreffenden Testanwendungsbeispiele als Testwegsammlung T bezeichnet werden können. Bei einem Testweg handelt es sich um eine Abfolge mutierter Meldung und entsprechender erwarteter Ausgabemeldungen. Unter den betreffenden mutierten Meldungen werden Eingabemeldungen verstanden, deren Felder einer Mutation unterzogen wurden. Die vorstehend bezeichnete Testwegsammlung T wird in die im Test befindliche Implementierung IUT eingegeben und durch die im Test befindliche Implementierung IUT erfolgt die Reaktion durch Ausgabemeldungen. Durch einen Vergleich zwischen den betreffenden Ausgabemeldungen und den der Testwegsammlung T entsprechenden erwarteten Ausgabemeldungen wird festgestellt, ob in der im Test befindlichen Implementierung IUT der Testwegsammlung T entsprechende Sicherheitslücken existieren. Mit diesem Verfahren kann für die im Test befindliche Implementierung IUT ein formalisiertes Spezifikationsmodell mit Störungsabdeckung bei Störungen bereitgestellt werden.

[0114] Durch die praktischen Ausführungsbeispiele der vorliegenden Erfindung erfolgt die Bereitstellung eines modellbasierten, formalisierten Protokollsicherheitstestverfahrens, wobei durch Einführung von Verhaltensmutationsanalyse, Modelltest, Syntaxmutationsanalyse und Syntaxtesttechnik in das Fuzz-Testing der Test von Sicherheitslücken erfolgt, welche von der im Test befindlichen Implementierung IUT unbekannten Lücken hervorgerufen werden. Nachstehend aufgeführt erfolgt die Beschreibung eines konkreten praktischen Ausführungsbeispiels des erfindungsgemäßen Protokollsicherheitstestverfahrens.

[0115] Bei [Abb. 2](#) handelt es sich um das Ablaufschema des durch die praktischen Ausführungsbeispiele der vorliegenden Erfindung bereitgestellten Protokollsicherheitstestverfahrens, welches die nachstehend aufgeführten Verfahrensschritte beinhaltet:

[0116] Verfahrensschritt **201**: Abbilden der der im Test befindlichen Implementierung IUT zugrunde liegenden Protokollspezifikationen zu einem ersten PEFSM (auch als ursprünglicher PEFSM bezeichnet).

[0117] Konkret wird vorgesehen, dass zur Erleichterung der Umwandlung der Protokollspezifikationen zu einem entsprechenden PEFSM durch den Anwender eine grafische Anwenderschnittstelle (UI) (beispielsweise Status und Übertragungsgrafik) sowie eine auf der erweiterten Kennzeichnungssprache (XML) basierende Statusübertragungssprache (beispielsweise grafische erweiterbare Kennzeichnungssprache (SCXML, State Chart XML) usw.) zur Beschreibung der verschiedenen Zustände des Protokolls sowie der entsprechenden Übertragungen und Eingabemeldungen/Ausgabemeldungen verwendet werden.

[0118] Unter Verwendung von Zuweisungsanweisungen oder anderen Operationen erfolgt die Expression der in dem Protokollstatus enthaltenen Berechnungen, einschließlich Eingabemeldungen (beziehungsweise darin enthaltene Felder) $i_k(\bar{v}_k)$, interne Variable $\bar{v}$ und Ausgabemeldungen (beziehungsweise darin enthaltene Felder), $o_k(\bar{w}_k)$.

[0119] [Abb. 3](#) zeigt den ursprünglichen PEFSM gemäß praktischem Ausführungsbeispiel. Wie in [Abb. 3](#) gezeigt ist s_0 , s_1 , s_2 der Protokollstatus. s_0 ist der ursprüngliche Protokollstatus und für die Übertragung zwischen den Zuständen erfolgt Kennzeichnung als i_k/O_k . Hierbei repräsentiert i_k die Eingabemeldungen, einschließlich Feldvektor $\bar{v}_k$, und löst die Übertragung aus. O_k repräsentiert die Ausgabemeldungen, einschließlich Feldvektor $\bar{w}_k$. Anschließend wird der betreffende PEFSM in einen neuen Zustand übertragen. Außerdem befindet sich in dem Feldvektor $\bar{v}_k$ ein TYPE-Bereich, welcher die Kategorie der Eingabemeldungen repräsentiert. Hierbei ist i_k , $0 \leq k \leq 4$, TYPE = t_k , beispielsweise ist für die Eingabemeldung i_0 der Wert des TYPE-Bereiches t_0 .

[0120] Verfahrensschritt **202**: In den durch Verfahrensschritt **201** erhaltenen PEFSM erfolgt die Einführung eines nicht ordnungsgemäßen Status, um den zweiten PEFSM zu erzeugen und somit in den ursprünglichen PEFSM eine Komponente einzuführen, welche Sicherheitsstörungen entspricht.

[0121] Bei dem vorliegenden praktischen Ausführungsbeispiel ist das Verfahren der Einführung des nicht ordnungsgemäßen Status in den ursprünglichen PEFSM wie nachstehend aufgeführt:

1) In den ursprünglichen PEFSM erfolgt die Eingabe eines nicht ordnungsgemäßen Status (auch als "worm-hole" bezeichnet), um die Modellierung einer der im Test befindlichen Implementierung IUT unbekannten Systemstörung und eines in den Definitionen der Protokollspezifikationen offensichtlich nicht definierten Status vorzunehmen. Außerdem werden hinsichtlich des nicht ordnungsgemäßen Status sämtliche interne Vektoren $\bar{v}$ und Felder der Ausgabemeldungen als Störungswerte eingestellt (dies bedeutet: nicht zu dem Bereich der durch die Protokollspezifikationen definierten ordnungsgemäßen Werte gehörige Werte). Wenn beispielsweise angenommen wird, dass der Bereich ordnungsgemäßer Werte für den internen Vektor $\bar{v}_i$ 0–16 beträgt, wird der Wert für den betreffende internen Vektor im nicht ordnungsgemäßen Status als kleiner 0 beziehungsweise als größer 16 eingestellt. Bei einem anderen Beispiel wird vorgesehen, dass das Feld TYPE der Eingabemeldung i_k als $\bar{t}$, $\bar{t} \neq t_k$, $0 \leq k \leq 4$ eingestellt wird.

2) Der betreffende nicht ordnungsgemäße Status weist eine externe Übertragung auf. Diese externe Übertragung kann mit der Kennzeichnung "Rücksetzung (reset)" versehen werden und verweist auf den ursprünglichen Status s_0 in der PEFSM. Diese mit der Kennzeichnung "reset" versehene Übertragung repräsentiert die Rücksetzungsoperation. Die betreffende Rücksetzungsoperation startet die im Test befindliche Implementierung IUT erneut und bewirkt, dass der entsprechende PEFSM aus den Systemstörungen beziehungsweise aus dem nicht definierten Status in den ursprünglichen Status s_0 zurückgesetzt wird. Weil der nicht ordnungsgemäße Status keineswegs zu den in den Protokollspezifikationen ursprünglich definierten Inhalten gehört, erfolgt keine Übertragung in den nicht ordnungsgemäßen Status. Dies bedeutet, dass keine Übertragung von dem ordnungsgemäßen ursprünglichen Status der PEFSM in den nicht ordnungsgemäßen Status möglich ist.

[0122] Nach der Einführung des nicht ordnungsgemäßen Status in den ursprünglichen PEFSM mittels vorstehend bezeichnetem Verfahren wird der zweite PEFSM erhalten wie in [Abb. 4](#) (ohne allerdings hierauf beschränkt zu sein) gezeigt: In [Abb. 4](#) wird ein nicht ordnungsgemäßer Status in den ursprünglichen PEFSM aus [Abb. 3](#) eingeführt. Dieser nicht ordnungsgemäße Status verfügt lediglich über eine Übertragung von dem ursprünglichen Status s_0 . Diese Übertragung wird mit der Kennzeichnung "reset" versehen und es kann keine Übertragung eines ordnungsgemäßen Status in dem ursprünglichen PEFSM zu dem betreffenden nicht ordnungsgemäßen Status erfolgen.

[0123] Verfahrensschritt **203**: Gemäß den bei der im Test befindlichen Implementierung IUT möglicherweise auftretenden Störungen erfolgt durch kategorische Festlegungen die Repräsentierung von Testzielen des ursprünglichen PEFSM und/oder von zweitem PEFSM und unter Verwendung von syntaktischen Beschränkungen erfolgt die Repräsentierung der syntaktischen Bestimmungen für ordnungsgemäße Eingabemeldungen von ursprünglichem PEFSM und/oder zweitem PEFSM.

[0124] Bei dem vorliegenden praktischen Ausführungsbeispiel werden Rechenbaumlogik (CTL) oder lineare temporale Logik (LTL) usw. als kategorische Festlegungen verwendet. Die betreffenden kategorischen Festlegungen werden verwendet, um entsprechend verschiedenen Sicherheitstestanforderungen konkrete Testziele zu repräsentieren. Bei dem folgenden Beispiel erfolgt die Verwendung einer CTL als kategorische Festlegung zur Repräsentierung der Testziele.

1) Entsprechend verschiedenen Sicherheitsanforderungen des Protokolls wie Authentifizierung (authentication), Autorisierung (authorization) usw. kann die Definition entsprechender Testziele vorgenommen werden. Beispielsweise kann gemäß der Definition der Protokollspezifikationen: s_1 als Authentifizierungsstatus, s_0 als ursprünglicher Status, die Funktion des betreffenden Protokolls lediglich nach ordnungsgemäßer Authentifizierung bereitgestellt werden. Die Definition des entsprechenden Testziels kann als nachstehend aufgeführte kategorische Festlegung erfolgen:

$$AXs_0 = s_1$$

[0125] Hierbei repräsentiert A "immer (always)" und X repräsentiert nächster (Next). Hierbei repräsentiert $AXs_0 = s_1$: In dem PEFSM muss der nächste Status von s_0 immer s_1 sein.

2) Gemäß dem im technischen Bereich der Sicherheitstests bekannten Wissen kann auch die Definition allgemeiner Testziele erfolgen.

[0126] Wenn beispielsweise angenommen wird, dass ein außergewöhnlicher Status "Wurmloch (wormhole)" (also ein nicht ordnungsgemäßer Status) in den ursprünglichen PEFSM eingeführt wird, um eine Systemstörung beziehungsweise einen nicht definierten Status zu repräsentieren, so gilt für eine korrekte im Test befindliche Implementierung IUT, dass für verschiedene Zustände der im Test befindlichen Implementierung IUT keine Übertragung in den Status "Wurmloch" erfolgt. Hierbei wird festgelegt, dass s_i die verschiedenen Zustände der ursprünglichen PEFSM repräsentiert, wobei $i = 0, 1, \dots$ ist und also mittels nachstehend aufgeführter kategorischer Festlegung ein allgemeines Testziel festgelegt wird:

$\text{!AG}(s_i \rightarrow \text{wormhole})$

[0127] Hierbei repräsentiert A "immer (always)", G repräsentiert "überall (globally)" und das Zeichen ! repräsentiert die Negation, so dass $\text{!AG}(s_i \rightarrow \text{wormhole})$ bedeutet: für einen beliebigen Status s_i erfolgt überall und immer keine Übertragung in den Status "Wurmloch".

3) Um sämtliche potentiellen Mängel in dem ursprünglichen PEFSM abdecken zu können, kann weiterhin vorgesehen werden, dass automatisch die Reflexion von kategorischen Festlegungen CTL aus dem ursprünglichen PEFSM als Testziele erfolgt. Bei den nachstehend aufgeführten Beispielen handelt es sich um praktische Verfahren zur Reflexion von kategorischen Festlegungen CTL aus dem ursprünglichen PEFSM. Allerdings besteht bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung keine Beschränkung auf diese Verfahren zur Reflexion von kategorischen Festlegungen CTL aus dem ursprünglichen PEFSM.

a. Hinsichtlich eines jeden Statuspaares s_i und s_j (ohne allerdings hierauf beschränkt zu sein) in dem ursprünglichen PEFSM erfolgt, sofern zwischen diesen keine Übertragung besteht, die Reflexion von zwei kategorischen Festlegungen:

$\text{!AG}(s_i \rightarrow s_j)$ und $\text{!AG}(s_j \rightarrow s_i)$

Durch die beiden genannten kategorischen Festlegungen wird eindeutig festgelegt, dass zwischen dem Status s_i und s_j keinerlei Übertragung existiert.

b. Hinsichtlich der Übertragung $s_i \rightarrow s_j$ in dem ursprünglichen PEFSM gilt, dass wenn in dem ursprünglichen PEFSM keine Übertragung $s_j \rightarrow s_i$ existiert, die Reflexion der nachstehend aufgeführten kategorischen Festlegung erfolgt:

$\text{!AG}(s_j \rightarrow s_i)$

Durch die vorstehend aufgeführte kategorische Festlegung wird festgelegt, dass in dem ursprünglichen PEFSM keine der Übertragung $s_i \rightarrow s_j$ entgegengesetzte Übertragung $s_j \rightarrow s_i$ existiert.

c. Hinsichtlich der Übertragung $s_i \rightarrow s_j$ in dem ursprünglichen PEFSM kann die Reflexion der nachstehend aufgeführten kategorischen Festlegung erfolgen:

$\text{AG}(s_i \rightarrow s_j)$

Durch die vorstehend aufgeführte kategorische Festlegung erfolgt die Festlegung, dass in dem ursprünglichen PEFSM eine ordnungsgemäße Übertragung $s_i \rightarrow s_j$ existiert.

d. Hinsichtlich jeder Übertragung bezüglich den Eingaben und Ausgaben in dem ursprünglichen PEFSM $s_i \xrightarrow{\text{in/out}} s_j$ kann die Reflexion der nachstehend aufgeführten kategorischen Festlegung erfolgen:

$\text{AG}((\text{state} = s_i \& \text{inPDU} = \text{in}) \rightarrow \text{AX}(\text{state} = s_j))$

[0128] Hierbei entsprechen in und out jeweils auf die betreffende Übertragung bezogenen Eingabemeldungen und Ausgabemeldungen.

[0129] Durch die vorstehend aufgeführte kategorische Festlegung erfolgt die Festlegung, dass in dem ursprünglichen PEFSM eine ordnungsgemäße Übertragung $s_i \xrightarrow{\text{in/out}} s_j$ existiert.

4) Abgesehen von den vorstehend aufgeführten kategorischen Festlegungen hinsichtlich von Status und Übertragung können auch kategorische Festlegungen hinsichtlich von internem Vektor $\bar{v}$, Feld der Eingabemeldung $\bar{v}_k$ und Aktion $A(\bar{V}, i_k(\bar{v}_k), o_k(\bar{w}_k))$ erzeugt werden.

[0130] Beispielsweise betragen die ordnungsgemäßen Werte des Feldes TYPE der Eingabemeldung i_k ($0 \leq k \leq 4$) wie nachstehend aufgeführt t_0, t_1, t_2, t_3 und t_4 , während andere Werte sämtlich nicht ordnungsgemäße

Werte darstellen. Somit kann die nachstehend aufgeführte kategorische Festlegung als Testziel verwendet werden:

$$AG(i_k.TYPE = t_0 \vee t_1 \vee t_2 \vee t_3 \vee t_4)$$

[0131] Durch die vorstehend aufgeführte kategorische Festlegung erfolgt die Festlegung, dass der Wert jeder beliebigen Eingabemeldung i_k überall und immer einer der Werte aus t_0, t_1, t_2, t_3 und t_4 ist. Hierbei wurden lediglich die beiden Operationssymbole AX und AG als Beispiel erläutert. Bei der tatsächlichen praktischen Verwendung können auch andere temporale logische Operationszeichen wie AF, AU, EX, EF, EG und EU als kategorische Festlegungen zur Definition der Testziele verwendet werden. Hierbei repräsentiert F "Zukunft (future)", U repräsentiert "bis (until)" und E repräsentiert "existiert (exist)".

[0132] Gleichzeitig können entsprechende syntaktische Beschränkungen automatisch aus den ordnungsgemäßen Eingabemeldungen des Formats XML aus ursprünglichem PEFSM und/oder zweitem PEFSM reflektiert werden und als XML-Muster verwendet werden. Die syntaktischen Merkmale (ohne allerdings hierauf beschränkt zu sein) des XML-Musters beinhalten:

- A. Position des Feldes in der Meldung;
- B. Anzahl des Auftretens des Feldes in der Meldung;
- C. Datenkategorie oder Code des Feldes in der Meldung usw.

[0133] Außerdem kann der Anwender entsprechend den Protokollspezifikationen weitere syntaktische Beschränkungen definieren beziehungsweise hinsichtlich des reflektierten XML-Musters manuelle Redaktionen, Abänderungen usw. vornehmen, beispielsweise (ohne allerdings hierauf beschränkt zu sein):

- D. Festlegung der Breite von Feldern fester Länge;
- E. Wertebereich der Felder usw.

[0134] Verfahrensschritt **204**: Durch Ausführung der Verhaltensmutationsoperation hinsichtlich des zweiten PEFSM mit eingeführtem nicht ordnungsgemäßigem Status erfolgt die Injektion der Sicherheitsstörung in den betreffenden zweiten PEFSM und es erfolgt die Erzeugung des mutierten PEFSM.

[0135] Bei dem vorliegenden praktischen Ausführungsbeispiel können die für die Ausführung der Verhaltensmutationsoperation verwendeten Verhaltensmutationsoperatoren Folgendes umfassen (ohne allerdings hierauf beschränkt zu sein): Statusfallenoperator (STO), Eingabeveränderungsoperator (IAO), Expressionsnegierungsoperator (ENO), Expressionsvermehrungsoperator (EIO), Expressionsverminderungsoperator (EDO), 0-Beibehaltungsoperator (stuck-at-0), 1-Beibehaltungsoperator (stuck-at-1), Operandennegierungsoperator (ONO), Operandenreferenzoperator (ORO), Bereichsexplorationsoperator (REO), Anwendungsbeispielinjektionsoperator (CIO) usw.

[0136] Bei dem vorliegenden praktischen Ausführungsbeispiel können die hinsichtlich der Übertragung $s_i \xrightarrow{in/out} s_j$ ausgeführten Mutationsoperationen die nachstehend aufgeführten Arten umfassen:

- 1) Unter Verwendung von Statusfallenoperator (STO) erfolgt die Ausführung der Mutationsoperation hinsichtlich der Übertragung $s_i \xrightarrow{in/out} wormhole$, zur Erzeugung der mutierten Übertragung $s_i \xrightarrow{in/out} s_j$, um den Status nach der Übertragung in wormhole zu verändern.
- 2) Unter Verwendung von Eingabeveränderungsoperator (IAO) erfolgt die Ausführung der Mutationsoperation hinsichtlich der Übertragung $s_i \xrightarrow{in/out} s_j$, zur Erzeugung der mutierten Übertragung $s_i \xrightarrow{in'/out} s_j$, wobei $in' \neq in$ ist. Dies bedeutet, dass die Eingabemeldungen (inPDU) der betreffenden Übertragungen nacheinander in andere Eingabemeldungen umgewandelt werden.

[0137] Für die Durchführung der Verhaltensmutationsoperation hinsichtlich der in dem Protokollstatus enthaltenen Zuweisungsanweisungen, einschließlich Eingabemeldungen (beziehungsweise darin enthaltene Felder) $i_k(\bar{v}_k)$, interne Variable $\bar{v}$ und Ausgabemeldungen (beziehungsweise darin enthaltene Felder), $o_k(\bar{w}_k)$ stehen die nachstehend aufgeführten Arten zur Verfügung:

- 1) Der Expressionsnegierungsoperator (ENO) wird verwendet, um eine Mutationsoperation der betreffenden Zuweisungsanweisung auszuführen und um den Expressionswert in der betreffenden Zuweisungsanweisung (auf der rechten Seite) (nach Stellen (bit)) umzukehren. Hierbei ist der Wert der Expression der Zuweisungsanweisung deren Berechnungsergebnis.
- 2) Der Expressionsvermehrungsoperator (EIO) wird verwendet, um eine Mutationsoperation der betreffenden Zuweisungsanweisung auszuführen und um den Expressionswert in der betreffenden Zuweisungsanweisung (auf der rechten Seite) um n zu vermehren.

- 3) Der Expressionsverminderungsoperator (EDO) wird verwendet, um eine Mutationsoperation der betreffenden Zuweisungsanweisung auszuführen und um den Expressionswert in der betreffenden Zuweisungsanweisung (auf der rechten Seite) um n zu vermindern.
- 4) Der 0-Beibehaltungsoperator (stuck-at-0) wird verwendet, um eine Mutationsoperation der betreffenden Zuweisungsanweisung auszuführen und um den Wert der Variablen in der betreffenden Zuweisungsanweisung (auf der linken Seite) auf 0 zu verändern, also sämtlich 0 für die Variable (auf der linken Seite) zuzuweisen.
- 5) Der 1-Beibehaltungsoperator (stuck-at-1) wird verwendet, um eine Mutationsoperation der betreffenden Zuweisungsanweisung auszuführen und um den Wert der Variablen in der betreffenden Zuweisungsanweisung (auf der linken Seite) auf 1 zu verändern, also FF für die Variable (auf der linken Seite) zuzuweisen.
- 6) Der Operandennegierungsoperator (ONO) wird verwendet, um eine Mutationsoperation der betreffenden Zuweisungsanweisung auszuführen und um die Operanden der Expression in der betreffenden Zuweisungsanweisung (auf der rechten Seite) umzukehren beziehungsweise nach Stellen umzukehren.
- 7) Falls in der Expression der betreffenden Zuweisungsanweisung (auf der linken Seite) mehrere Operanden vorhanden sind, was Berechnungen mehrerer Zahlenwerte bedeutet, wird der Operandenreferenzoperator (ORO) verwendet, um eine Mutationsoperation der betreffenden Zuweisungsanweisung auszuführen und um die Operanden nacheinander mit ihrem nachfolgenden Operanden auszutauschen und den letzten Operanden mit dem ersten Operanden auszutauschen.
- 8) Der Bereichsexplorationoperator (REO) wird verwendet, um eine Mutationsoperation der betreffenden Zuweisungsanweisung auszuführen und um den Wert von jedem Operanden der Expression in der betreffenden Zuweisungsanweisung (auf der rechten Seite) nach Stellen umzukehren (also wie bei ENO) und die betreffende Zuweisungsanweisung mit der Kennzeichnung "fuzz:replace:n" zu versehen, wobei es sich bei n um die Bytebreite des Feldes handelt. Unter Feld wird hierbei die auf der linken Seite der Zuweisungsanweisung befindliche Variable verstanden. n beträgt normalerweise 1, 2, 4 oder 8 Byte.
- 9) Falls die Zuweisungsanweisung Meldungsfelder variabler Länge beinhaltet und die Länge der entsprechenden Felder betreffende kategorische Festlegungen existieren, kann der Anwendungsbeispielsinjektionsoperator (CIO) verwendet werden, um eine Mutationsoperation der betreffenden Zuweisungsanweisung auszuführen, wobei der Anwendungsbeispielsinjektionsoperator die Länge des betreffenden Feldes variabler Länge um n Byte vermehrt und die betreffende Zuweisungsanweisung mit der Kennzeichnung "fuzz:inject" versieht.

[0138] Auf diese Weise kann hinsichtlich von jeder in dem zweiten PEFSM enthaltenen Übertragung $s_i \rightarrow s_j$ sowie deren interner Variablen und Feldern der Eingabemeldungen/Ausgabemeldungen durch Ausführung der Verhaltensmutationsoperation der entsprechende mutierte PEFSM erhalten werden, wobei jeder verwendete Verhaltensmutationsoperator (beispielsweise Operandenreferenzoperator, Expressionsnegierungsoperator) eine Mutationsoperation zur Erzeugung eines mutierten PEFSM ausführt und wobei die betreffenden mutierten PEFSM lediglich eine Sicherheitsstörung beinhalten und von dem ursprünglichen PEFSM abweichen. Wenn beispielsweise der verwendete Operandenreferenzoperator hinsichtlich der in [Abb. 4](#) gezeigten Übertragung $s_2 \rightarrow s_0$ in dem PEFSM (es handelt sich um eine Übertragung in dem in [Abb. 3](#) gezeigten ursprünglichen PEFSM) eine Mutationsoperation ausführt, um die betreffende Übertragung $s_2 \rightarrow s_0$ zu der Übertragung $s_2 \rightarrow s_{\text{nicht ordnungsgemäßer Status}}$ zu mutieren, so wird der in [Abb. 4](#) gezeigte zweite PEFSM zu dem in [Abb. 5](#) gezeigten mutierten PEFSM mutiert.

[0139] Verfahrensschritt **205**: hinsichtlich des mutierten PEFSM erfolgt die Ausführung des Modelltests zur Erzeugung von einem den in Verfahrensschritt **203** erzeugten kategorischen Festlegungen widersprechenden Gegenbeispiel oder einer Mehrzahl von solchen Gegenbeispielen in diesem mutierten PEFSM, wobei dieses eine Gegenbeispiel oder eine Mehrzahl von solchen Gegenbeispielen als Testabfolge abgebildet wird.

[0140] Bei dem vorliegenden praktischen Ausführungsbeispiel wird die NuSMV-Modellüberprüfungsvorrichtung als zur Erzeugung der Testabfolgen verwendet, wobei der vorstehend bezeichnete Verfahrensschritt **205** Folgendes beinhaltet:

- 1) Der in Verfahrensschritt **204** erhaltene mutierte PEFSM wird durch die NuSMV-Sprache in das NuSMV-Modell übersetzt, wobei die NuSMV-Zuweisungsbeschränkung (NuSMV ASSIGN constraint) für die Repräsentierung der Statusübertragung in dem PEFSM verwendet wird. Bei dem vorliegenden praktischen Ausführungsbeispiel kann ein mutierter PEFSM zu dem nachstehend aufgeführten NuSMV-Modell übersetzt werden:

```
MODULE main
VAR
state      : {s0, s1, s2, wormhole};
input      : { I0, I1, I2, I3, I4, reset};
TYPE :{ t0, t1, t2, t3, t4,  $\bar{t}$ }
output     :{ null, O0, O1, O2, O3, O4};
ASSIGN
init(state) := s0;
next(state) :=
case
    state = s0 & input= I0: s0;
    state = s0 & input= I1: s1;
    state = s1 & input= I2: s2;
```

```

state =  $s_2$  & input=  $I_3$ :  $s_2$ ;
state =  $s_2$  & input=  $I_4$ : wormhole;
state = wormhole:  $s_0$ ;
1 : state;
esac;

init(output) := null;
next(output) :=
case
    input =  $I_0$  :  $O_0$ ;
    input =  $I_1$  :  $O_1$ ;
    input =  $I_2$  :  $O_2$ ;
    input =  $I_3$  :  $O_3$ ;
    input =  $I_4$  :  $O_4$ ;
    input = reset : null;
    1 : null;
esac;

init(input) := {  $I_0$ ,  $I_1$  };
next(input) :=
case
    state =  $s_0$  & input= {  $I_0$ ,  $I_1$  };
    state =  $s_0$  & input=  $I_1$ :  $I_1$ ;
    state =  $s_1$  & input=  $I_2$ : {  $I_3$ ,  $I_4$  };
    state =  $s_2$  & input=  $I_3$ : {  $I_3$ ,  $I_4$  };
    state =  $s_2$  & input=  $I_4$ : reset;
    state = wormhole & input= reset : {  $I_0$ ,  $I_1$  };
    1 : null;

```

```

esac;

init(TYPE) := null;

next(TYPE) :=

case

    state = s0 & input= I0      : t0;

    state = s0 & input= I1      : t1;

    state = s1 & input= I2      : t2;

    state = s2 & input= I3      : t3;

    state = s2 & input= I4      : t4;

    state = wormhole      :  $\bar{t}$ ;

    1 : null;

esac;

```

2) Die NuSMV-Modellüberprüfungsvorrichtung führt den Modelltest hinsichtlich der kategorischen Festlegungen des in Verfahrensschritt **204** erzeugten mutierten PEFSM und der in Verfahrensschritt **203** zur Verwendung als Repräsentierung von Testzielen erzeugten kategorischen Festlegungen aus, um in dem mutierten PEFSM enthaltene, den kategorischen Festlegungen widersprechende Testanwendungsbeispiele (auch als Gegenbeispiele bezeichnet) festzustellen. Wenn die NuSMV-Modellüberprüfungsvorrichtung beispielsweise feststellt, dass der PEFSM den kategorischen Festlegungen $\neg \text{AG}(s_i \rightarrow \text{wormhole})$ und $\text{AG}(i_k.\text{TYPE} = t_0 \vee t_1 \vee t_2 \vee t_3 \vee t_4)$ widerspricht, erfolgt die Erzeugung eines entsprechenden Gegenbeispiels.

3) Konkret wird vorgesehen, dass, da das erzeugte Gegenbeispiel einem Übertragungsweg entspricht, in dem vorliegenden praktischen Ausführungsbeispiel das Gegenbeispiel als Kennzeichnung der auf diesem Übertragungsweg existierenden Eingabemeldungen/Ausgabemeldungen dargestellt wird. Hierbei erfolgt Abbilden der entsprechenden Kennzeichnung zu den entsprechenden Eingabemeldungen/Ausgabemeldungen. Die von diesen Eingabemeldungen/Ausgabemeldungen gebildeten Meldungsabfolgen stellen eine Testabfolge dar. Daher können Gegenbeispiele unmittelbar zu tatsächlichen Testabfolgen (testing traces) abgebildet werden. Die betreffenden Testabfolgen beinhalten: eine Reihe von Eingabemeldungen zur Provokation der im Test befindlichen Implementierung IUT und Ausgabemeldungen, welche den erwarteten Reaktionen der im Test befindlichen Implementierung IUT entsprechen.

[0141] Gleichzeitig erfolgt die Kennzeichnung hinsichtlich von jeder beliebigen Meldung, welche einer den kategorischen Festlegungen widersprechenden mutierten Übertragung (beispielsweise der Übertragung von dem ordnungsgemäßen Status zu dem nicht ordnungsgemäßen Status) entspricht, sowie der von der Modellüberprüfungsvorrichtung identifizierten Felder, wobei das Versehen mit Kennzeichnungen der betreffenden Meldungen und Felder gemäß dem nachstehend aufgeführten Verfahren durchgeführt werden kann:

- Falls die Modellüberprüfungsvorrichtung eine Identifizierung dahingehend vornehmen kann, dass die Mutation eines gewissen Feldes ein Gegenbeispiel herbeiführen kann und das betreffende Feld über keine andere Kennzeichnung verfügt, erfolgt die Kennzeichnung des betreffenden Feldes mit "fuzz (Fuzz-Testing)"; anderenfalls erfolgt die Kennzeichnung der gesamten Meldung mit "fuzz".
- Falls die dem identifizierten Feld entsprechende Zuweisungsanweisung mit der Kennzeichnung "fuzz: replace:n" versehen ist, erfolgt die Kennzeichnung des betreffenden Feldes in der Meldung mit "replace" und es erfolgt die Durchführung von Fuzz-Testing für das betreffende Feld unter Verwendung einer Testabfolge unter Verwendung der Länge n.
- Falls die dem identifizierten Feld entsprechende Zuweisungsanweisung mit der Kennzeichnung "fuzz: inject" versehen ist, erfolgt die Kennzeichnung des betreffenden Feldes in der Meldung mit "inject" und es erfolgt die Durchführung von Fuzz-Testing für das betreffende Feld unter Verwendung sämtlicher Testabfolgen.

[0142] Weiterhin wird vorgesehen, dass zwecks Steigerung der Testeffizienz vor der Durchführung des Modelltests und/oder nach der Durchführung des Modelltests eine Optimierung der Testanwendungsbeispiele vorgenommen wird, um darin enthaltene redundante Testanwendungsbeispiele zu entfernen.

[0143] Verfahrensschritt **206**: Ausführung der Syntaxmutationsoperation hinsichtlich der vorstehend bezeichneten Testabfolgen und Injektion syntaktischer Störungen in die betreffenden Testabfolgen zur Erzeugung von jeder Testabfolge entsprechenden mutierten Testabfolgen. Hierbei entspricht eine Testabfolge einer mutierten Meldungen enthaltenden mutierten Testabfolge oder einer Mehrzahl von solchen Testabfolgen.

[0144] Hinsichtlich der in Verfahrensschritt **205** erzeugten Testabfolgen erfolgt die Ausführung der Syntaxmutationsoperation. Konkret wird vorgesehen, dass hinsichtlich jeder mit der Kennzeichnung "fuzz" versehene Meldung in der im Modelltest erzeugten Testabfolge beziehungsweise hinsichtlich der mit der Kennzeichnung "fuzz" versehene Felder enthaltenden Meldungen unter Verwendung eines beliebigen der nachstehend aufgeführten Syntaxmutationsoperatoren (ohne allerdings hierauf beschränkt zu sein) oder einer beliebigen Kombination der nachstehend aufgeführten Syntaxmutationsoperatoren: Feldlöschungsoperator (Field Erase Operator), Feldwiederholungsoperator (Field Repeat Operator), Feldaustauschoperator (Field Exchange Operator), Breitenveränderungsoperator (Width Change Operator), Codeveränderungsoperator (Encode Change Operator) und Wertnegierungsoperator (Value Negation Operator) die Durchführung der vorstehend bezeichneten Syntaxmutationsoperation der vorstehend bezeichneten Meldungen erfolgt, um entsprechende mutierte Meldungen zu erzeugen.

[0145] Hierbei wird der Feldlöschungsoperator verwendet, um entsprechende mit der Kennzeichnung "fuzz" versehene Felder in den Meldungen zu löschen.

[0146] Der Feldwiederholungsoperator wird verwendet, um mit der Kennzeichnung "fuzz" versehene Felder in den Meldungen zu n Kopien zu kopieren. Der Feldaustauschoperator wird verwendet, um die Position von aktuell mit der Kennzeichnung "fuzz" versehenen Feldern mit der Position des nachfolgenden Feldes auszutauschen.

[0147] Der Breitenveränderungsoperator wird verwendet, um den Inhalt von aktuell mit der Kennzeichnung "fuzz" versehenen Feldern um n Byte zu vermehren, wobei es sich bei n um eine negative ganze Zahl handeln kann.

[0148] Der Codeveränderungsoperator wird verwendet, um das Codeformat von mit der Kennzeichnung "fuzz" versehenen Feldern zu einem anderen Format zu verändern, beispielsweise von ASCII zur UTF-8, UNICODE usw. Der Wertnegierungsoperator wird verwendet, um den Wert von mit der aktuellen Kennzeichnung "fuzz" versehenen Feldern umzukehren beziehungsweise nach Stellen umzukehren.

[0149] Verfahrensschritt **207**: Hinsichtlich der in den mutierten Testabfolgen enthaltenen mutierten Meldungen erfolgt die Durchführung von Syntaxtests, um die in den vorstehend bezeichneten mutierten Meldungen enthaltenen gültigen Meldungen zu erhalten, welche den syntaktischen Beschränkungen widersprechen. Die vorstehend bezeichnete gültige mutierte Meldungen enthaltenden mutierten Testabfolgen werden als Testanwendungsbeispiele verwendet.

[0150] Es erfolgt die Verwendung des entsprechenden XML-Musters, also der syntaktischen Beschränkung, zur Durchführung des Syntaxtests hinsichtlich der vorstehend bezeichneten mutierten Meldungen, wobei den syntaktischen Beschränkungen in dem XML-Muster widersprechende mutierte Meldungen als gültige mutierte Meldungen verwendet werden. Die vorstehend bezeichnete gültige mutierte Meldungen enthaltenden Testabfolgen werden als Testanwendungsbeispiele für die folgende Durchführung von Fuzz-Testing verwendet. Hierbei gelten den syntaktischen Beschränkungen in dem XML-Muster nicht widersprechende mutierte Meldungen als nicht gültige mutierte Meldungen, wobei solche Meldungen ihrem Wesen nach ordnungsgemäße Meldungen sind. Aus diesem Grund können solche nicht gültige mutierte Meldungen enthaltende Testabfolgen entfernt werden.

[0151] Verfahrensschritt **208**: Unter Verwendung der in Verfahrensschritt **207** enthaltenen Testanwendungsbeispiele erfolgt die Durchführung von Fuzz-Testing hinsichtlich der im Test befindlichen Implementierung IUT, um zu testen, ob in der betreffenden im Test befindlichen Implementierung IUT entsprechende Sicherheitslücken existieren.

[0152] Konkret wird vorgesehen, dass der Protokollsicherheitstest hinsichtlich eines Testanwendungsbeispiels wie nachstehend aufgeführt durchgeführt wird:

1) Hinsichtlich von in den Eingabemeldungen enthaltenen mit der Kennzeichnung "replace" beziehungsweise "inject" versehenen Feldern erfolgt Ersetzung durch den vorab definierten Testwert beziehungsweise durch den der kategorischen Festlegung widersprechenden nicht ordnungsgemäßen Wert oder es erfolgt Injektion in das Feld der betreffenden Eingabemeldung, um eine mutierte Meldung zur Verwendung bei dem Fuzz-Testing zu erzeugen.

[0153] Hierbei wird vorgesehen, dass es sich bei dem der kategorischen Festlegung widersprechenden nicht ordnungsgemäßen Wert um einen außerhalb des Bereiches der ordnungsgemäßen Werte ausgewählten Wert beziehungsweise um einen kollidierenden Wert beziehungsweise um einen zufällig erzeugten Wert oder um einen vorab definierten Wert beziehungsweise um einen auf andere Weise erzeugten Wert handelt. Außerdem können der kategorischen Festlegung widersprechende Werte manchmal durch Gegenbeispiele und/oder mutierten PEFSM ausgegeben werden.

2) An die im Test befindliche Implementierung IUT erfolgt nacheinander die Versendung von jeder in den Testanwendungsbeispielen enthaltenen Eingabemeldung.

3) Hinsichtlich der von der im Test befindlichen Implementierung IUT empfangenen Reaktionsmeldungen wird vorgesehen, dass hinsichtlich jeder empfangenen Reaktionsmeldung die entsprechende Reaktionsmeldung mit den in den Testanwendungsbeispielen enthaltenen erwarteten Ausgabemeldungen verglichen wird, um zu testen, ob die entsprechende Reaktionsmeldung mit der in dem Testanwendungsbeispiel enthaltenen Ausgabemeldung übereinstimmt. Falls die getesteten Reaktionsmeldungen nicht mit den erwarteten Ausgabemeldungen übereinstimmen, kann eine Beurteilung dahingehend vorgenommen werden, dass bei der im Test befindlichen Implementierung IUT möglicherweise eine entsprechende Sicherheitslücke existiert.

4) Nach Beendigung der Versendung der vorstehend bezeichneten Testmeldungen und des vorstehend bezeichneten Vergleichs erfolgt die Versendung von ordnungsgemäßen Versendungsexplorationsmeldungen an die im Test befindliche Implementierung IUT, um zu testen, ob bei der im Test befindlichen Implementierung IUT eine Systemstörung oder ein nicht definierter Status vorliegt. Falls sich die im Test befindliche Implementierung IUT in nicht ordnungsgemäßen Status befindet, wird eine Sicherheitslücke festgestellt und das aktuelle Testanwendungsbeispiel kann als Missbrauchsscript für die Feststellung von Sicherheitslücken dienen.

[0154] Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung wird vorgesehen, dass bei den vorstehend bezeichneten Verfahrensschritten **204** bis **208** der Interaktivmodus verwendet wird. Es wird vorgesehen, dass jedes Mal, wenn bei Verfahrensschritt **204** durch Verhaltensmutationsoperation ein mutiertes Protokollstatusmodell erhalten wird und durch Verfahrensschritt **205** hinsichtlich dieses mutierten Protokollstatusmodells die Durchführung des Modelltests erfolgt, um ein Gegenbeispiel oder eine Mehrzahl von solchen Gegenbeispielen zu erzeugen, in Verfahrensschritt **206** durch Ausführung der Syntaxmutationsoperation die jedem Gegenbeispiel entsprechende mutierte Testabfolge oder eine Mehrzahl von solchen mutierten Testabfolgen erhalten wird. In Verfahrensschritt **207** erfolgt die Durchführung des Syntaxtests hinsichtlich der vorstehend bezeichneten mutierten Testabfolge oder einer Mehrzahl von solchen mutierten Testabfolgen. Bei der Festlegung eines Testanwendungsbeispiels hieraus erfolgt in Verfahrensschritt **208** die Verwendung des betreffenden Testanwendungsbeispiels zur Durchführung von Fuzz-Testing. Nach der Durchführung von Fuzz-Testing hinsichtlich sämtlicher Testanwendungsbeispiele von jedem Gegenbeispiel aus Verfahrensschritt **205** erfolgt Rückkehr zu Verfahrensschritt **204** mit Festlegung einer neuen Verhaltensmutationsoperation zur Festlegung eines mutierten Protokollstatusmodells. Es erfolgt die Wiederholung der vorstehend bezeichneten Operation. Nach Durchführung von Fuzz-Testing hinsichtlich sämtlichen Gegenbeispielen sämtlicher Protokollstatusmodelle entsprechender Testanwendungsbeispiele erfolgt die Beendigung.

[0155] Außerdem kann vorgesehen werden, dass jedes Mal, wenn in Verfahrensschritt **204** durch Verhaltensmutationsoperation ein mutiertes Protokollstatusmodell erhalten wird und in Verfahrensschritt **205** hinsichtlich dieses mutierten Protokollstatusmodells die Durchführung des Modelltests erfolgt, eine Testabfolge erzeugt werden kann. Weiterhin erfolgt in Verfahrensschritt **206** durch Ausführung der Syntaxmutationsoperation das Erhalten einer mutierten Testabfolge. Während der Festlegung der betreffenden mutierten Testabfolge als Testanwendungsbeispiel in Verfahrensschritt **207** erfolgt in Verfahrensschritt **208** die Verwendung des betreffenden Testanwendungsbeispiels, um mit der Kennzeichnung "replace" oder "inject" versehene Felder auszutauschen oder die Injektion verschiedener nicht ordnungsgemäßer Werte zur Durchführung von Fuzz-Testing vorzunehmen. Deshalb verhält es sich so, dass nach der erfolgten Durchführung von Fuzz-Testing unter Verwendung des aktuellen Testanwendungsbeispiels die Rückkehr zu Verfahrensschritt **206** zur Erzeugung der nächsten mutierten Testabfolge erfolgt. In Verfahrensschritt **207** erfolgt die Durchführung des Syntaxtests hin-

sichtlich der vorstehend bezeichneten mutierten Testabfolge. Bei der Festlegung der betreffenden mutierten Testabfolge als Testanwendungsbeispiel erfolgt in Verfahrensschritt **208** die Durchführung von Fuzz-Testing. Dieser Vorgang wird wiederholt, bis die Erzeugung sämtlicher möglicher Testanwendungsbeispiele sämtlicher Testabfolgen aus Verfahrensschritt **205** erfolgt ist sowie deren Test durchgeführt worden ist. Es erfolgt Rückkehr zur Verfahrensschritt **204** zur Erzeugung des nächsten mutierten Protokollstatusmodells und zur Wiederholung des vorstehend bezeichneten Vorgangs, bis die Erzeugung sämtlicher Testanwendungsbeispiele sämtlicher Testabfolgen der vorstehend bezeichneten mutierten Protokollstatusmodelle erfolgt ist und der Test beendet ist.

[0156] Basierend auf dem vorstehend bezeichneten Protokollsicherheitstestverfahren erfolgt durch die praktischen Ausführungsbeispiele der vorliegenden Erfindung außerdem die Bereitstellung einer Vorrichtung zur Durchführung von Sicherheitstests von Protokollimplementierungen. Wie in [Abb. 6](#) gezeigt umfasst die betreffende Vorrichtung:

ein erstes Modul **601**, welches verwendet wird, um während der Durchführung des Sicherheitstests hinsichtlich der vorstehend bezeichneten Protokollimplementierung das Abbilden der vorstehend bezeichneten Protokollimplementierung zugrunde liegenden Protokollspezifikationen zu einem ersten Protokollstatusmodell vorzunehmen;

ein zweites Modul **602**, welches verwendet wird, um das von dem vorstehend bezeichneten ersten Modul **601** erhaltene vorstehend bezeichnete erste Protokollstatusmodell in einen nicht ordnungsgemäßen Status einzuführen und das zweite Protokollstatusmodell zu erzeugen, wobei der vorstehend bezeichnete nicht ordnungsgemäße Status bei der vorstehend bezeichneten Protokollimplementierung existierenden Sicherheitsstörungen entspricht;

ein drittes Modul **603**, welches verwendet wird, um unter Verwendung kategorischer Festlegungen die Repräsentierung von Testzielen des von vorstehend bezeichnetem zweiten Modul **602** stammenden vorstehend bezeichneten zweiten Protokollstatusmodells und/oder des von vorstehend bezeichnetem ersten Modul **601** stammenden vorstehend bezeichneten ersten Protokollstatusmodells vorzunehmen, wobei unter Verwendung syntaktischer Beschränkungen die Repräsentierung der syntaktischen Bestimmungen ordnungsgemäßer Eingabemeldungen des vorstehend bezeichneten zweiten Protokollstatusmodells und/oder des vorstehend bezeichneten ersten Protokollstatusmodells erfolgt;

ein viertes Modul **604**, welches verwendet wird, um hinsichtlich des von vorstehend bezeichnetem zweiten Modul **602** stammenden vorstehend bezeichneten zweiten Protokollstatusmodells die Verhaltensmutationsoperation vorzunehmen und auf diese Weise das mutierte Protokollstatusmodell zu erzeugen;

ein fünftes Modul **605**, welches verwendet wird, um unter Verwendung der von vorstehend bezeichnetem dritten Modul **603** erhaltenen kategorischen Festlegungen hinsichtlich des von vorstehend bezeichnetem vierten Modul **604** erzeugten vorstehend bezeichneten mutierten Protokollstatusmodells den Modelltest auszuführen und eine Testabfolge oder eine Mehrzahl von solchen Testabfolgen zu erzeugen, wobei die vorstehend bezeichnete Testabfolge oder eine Mehrzahl von solchen Testabfolgen durch Abbilden des in dem vorstehend bezeichneten mutierten Protokollstatusmodell enthaltenen, den vorstehend bezeichneten kategorischen Festlegungen widersprechenden Gegenbeispiels oder einer Mehrzahl von solchen Gegenbeispielen erhalten wird;

ein sechstes Modul **606**, welches verwendet wird, um hinsichtlich der von vorstehend bezeichnetem fünften Modul **605** stammenden Testabfolge die Syntaxmutationsoperation vorzunehmen und die mutierte Meldungen enthaltende mutierte Testabfolge zu erzeugen, wobei eine Testabfolge eine Mutationen enthaltende mutierte Testabfolge oder eine Mehrzahl von solchen Mutationen enthaltenden Testabfolgen erzeugt;

ein siebtes Modul **607**, welches verwendet wird, um unter Verwendung der von vorstehend bezeichneten dritten Modul **603** erhaltenen syntaktischen Beschränkungen hinsichtlich der von vorstehend bezeichnetem sechsten Modul **606** erzeugten mutierten Testabfolgen den Syntaxtest durchzuführen und die in den vorstehend bezeichneten mutierten Meldungen enthaltenen, den vorstehend bezeichneten syntaktischen Beschränkungen widersprechenden gültigen mutierten Meldungen zu erhalten, wobei die vorstehend bezeichnete gültige mutierte Meldungen enthaltenden mutierten Testabfolgen als Testanwendungsbeispiele verwendet werden; sowie

ein achttes Modul **608**, welches verwendet wird, um unter Verwendung der von vorstehend bezeichnetem siebten Modul **607** erhaltenen Testanwendungsbeispiele hinsichtlich der Protokollimplementierung die Durchführung von Fuzz-Testing vorzunehmen, um zu testen, ob bei der vorstehend bezeichneten Protokollimplementierung einem Testanwendungsbeispiel oder einer Mehrzahl von solchen Testanwendungsbeispielen entsprechende Sicherheitslücken existieren. Bei der konkreten Umsetzung wird vorgesehen, dass – entsprechend dem in [Abb. 1](#) gezeigten Verfahren – das vorstehend bezeichnete fünfte Modul **605** die Ausführung der ersten Variante der Syntaxmutation vornimmt. Hinsichtlich der von dem fünften Modul **605** erzeugten und lediglich mit den Kennzeichnungen “replace” oder “inject” versehene Felder enthaltenden Testabfolgen, jedoch nicht hinsichtlich von mit der Kennzeichnung “fuzz” versehene Felder enthaltenden Testabfolgen, ist keine Behandlung durch das sechste Modul **606** erforderlich. Vielmehr erfolgt die unmittelbare Ausgabe als Testanwendungsbeispiele an das achte Modul **608** zur Durchführung von Fuzz-Testing. Hierbei kann das achte Modul **608** auch

von dem fünften Modul **605** stammende Testanwendungsbeispiele zur Durchführung des Modelltests der Protokollimplementierung verwenden.

[0157] Bevorzugt wird vorgesehen, dass das fünfte Modul **605** Abbilden von jedem Gegenbeispiel zu einer Testabfolge vornimmt.

[0158] Konkret wird vorgesehen, dass zwischen dem vierten Modul **604**, dem fünften Modul **605**, dem sechsten Modul **606**, dem siebten Modul **607** und dem achten Modul **608** zwei Betriebsweisen vorgesehen werden können:

I. Gruppenbehandlungsmodus

[0159] Konkret werden verschiedene Arten des Gruppenbehandlungsmodus vorgesehen, wovon nachstehend zwei Arten beispielhaft aufgeführt werden:

1) Wenn das vierte Modul **604** bereits sämtliche mutierte Protokollstatusmodelle erzeugt hat, das fünfte Modul **605** bereits durch Ausführung des Modelltests sämtliche Testabfolgen hinsichtlich sämtlicher mutierter Protokollstatusmodelle erhalten hat, das sechste Modul **606** hinsichtlich sämtlicher Testabfolgen die Erzeugung mutierter Testabfolgen vorgenommen hat und das siebte Modul **607** die Festlegung sämtlicher Testanwendungsbeispiele vorgenommen hat, erfolgt durch das achte Modul **608** im Gruppenbehandlungsmodus die Durchführung von Fuzz-Testing.

2) Viertes Modul **604** und siebtes Modul **607** werden parallel ausgeführt. Wenn das vierte Modul **604** durch Ausführung der Verhaltensmutationsoperation ein mutiertes Protokollstatusmodell erhalten hat und das fünfte Modul **605** durch Modelltest hinsichtlich des betreffenden mutierten Protokollstatusmodells ein Gegenbeispiel oder eine Mehrzahl von solchen Gegenbeispielen erzeugt hat, erfolgt durch das sechste Modul **606** die Durchführung der Syntaxmutationsoperation zum Erhalten der jedem Gegenbeispiel entsprechenden mutierten Testabfolge. Das siebte Modul **607** führt den Syntaxtest hinsichtlich der mutierten Testabfolge durch, um die Festlegung sämtlicher Testanwendungsbeispiele vorzunehmen, bis sämtliche Testanwendungsbeispiele sämtlicher Testabfolgen sämtlicher Protokollstatusmodelle erzeugt worden sind. Anschließend erfolgt durch das achte Modul **608** nacheinander die Durchführung von Fuzz-Testing hinsichtlich sämtlicher erzeugter Testanwendungsbeispiele.

II. Interaktivmodus (Interactive mode)

[0160] Konkret werden verschiedene Arten des Interaktivmodus vorgesehen, wovon nachstehend zwei Arten beispielhaft aufgeführt werden:

1) Es wird vorgesehen, dass jedes Mal, wenn das vierte Modul **604** ein mutiertes Protokollstatusmodell erzeugt hat, das fünfte Modul **605** durch Ausführung des Modelltests hinsichtlich des betreffenden mutierten Protokollstatusmodells eine diesem mutierten Protokollstatusmodell entsprechende Testabfolge oder eine Mehrzahl von solchen Testabfolgen erhalten hat, das sechste Modul **606** die Syntaxmutationsoperation ausführt, um die jedem mutierten Gegenbeispiel entsprechende mutierte Testabfolge zu erhalten. Das siebte Modul **607** führt hinsichtlich der vorstehend bezeichneten mutierten Testabfolge den Syntaxtest zur Festlegung der Testanwendungsbeispiele durch. Das achte Modul **608** führt unter Verwendung von einem dem betreffenden Gegenbeispiel entsprechenden Testanwendungsbeispiel oder einer Mehrzahl von solchen Testanwendungsbeispielen das Fuzz-Testing durch.

2) Es wird vorgesehen, dass jedes Mal, wenn das vierte Modul **604** ein mutiertes Protokollstatusmodell erzeugt hat, das fünfte Modul **605** durch Ausführung des Modelltests hinsichtlich des betreffenden mutierten Protokollstatusmodells ein Gegenbeispiel erzeugt. Weiterhin erfolgt mittels Ausführung der Syntaxmutationsoperation durch das sechste Modul **606** der Erhalt einer mutierten Testabfolge und das siebte Modul **607** führt hinsichtlich der mutierten Testabfolge den Syntaxtest durch zur Festlegung der betreffenden mutierten Testabfolge als Testanwendungsbeispiel. Durch das achte Modul **608** erfolgt unter Verwendung des betreffenden Testanwendungsbeispiels die Durchführung von Fuzz-Testing.

[0161] Optional kann vorgesehen werden, dass das dritte Modul **603** nach Erhalten der die Testziele repräsentierenden kategorischen Festlegungen eine weitere Optimierung der betreffenden kategorischen Festlegungen vornimmt, um darin enthaltene redundante kategorische Festlegungen zu entfernen.

[0162] Optional kann vorgesehen werden, dass das fünfte Modul **605** nach Erhalten der Testabfolgen eine weitere Optimierung der betreffenden Testabfolgen vornimmt, um darin enthaltene redundante Testabfolgen zu entfernen.

[0163] Optional kann vorgesehen werden, dass die vorstehend bezeichnete Vorrichtung zur Durchführung von Protokollsicherheitstests auch einen Konsistenztest der Protokollimplementierung durchführt. Hierbei kann das vierte Modul **604** hinsichtlich der durch das dritte Modul **603** erhaltenen kategorischen Festlegungen die Mutationoperation vornehmen, um die mutierten kategorischen Festlegungen zu erzeugen. Das fünfte Modul **605** kann unter Verwendung der durch das vierte Modul **604** erzeugten mutierten kategorischen Festlegungen hinsichtlich des durch das erste Modul **601** erhaltenen ersten Protokollstatusmodells den Modelltest vornehmen, um das zur Verwendung bei dem Konsistenztest verwendete Testanwendungsbeispiel oder eine Mehrzahl von solchen Testanwendungsbeispielen zu erhalten. Das sechste Modul **606** kann weiterhin unter Verwendung des durch das fünfte Modul **605** erhaltenen, für die Durchführung des Konsistenztests verwendeten Testanwendungsbeispiels oder einer Mehrzahl von solchen Testanwendungsbeispielen hinsichtlich der betreffenden Protokollimplementierung den Konsistenztest vornehmen und mit der Protokollspezifikation nicht übereinstimmende Sicherheitslücken feststellen.

[0164] Die vorstehend bezeichneten verschiedenen Verfahren zur Ausführung der verschiedenen Funktionen durch die vorstehend bezeichneten Module sind bereits detailliert beschrieben worden, so dass an dieser Stelle eine wiederholte Erläuterung unterbleiben kann.

[0165] Außerdem kann die vorstehend bezeichnete Vorrichtung zur Durchführung der Protokollsicherheitstests als Hardware-Anlage verwirklicht werden. Auch ist eine Verwirklichung als Simulationsvorrichtung in Form von Software möglich, wobei die betreffende Simulationsvorrichtung in einer Computervorrichtung zwecks Durchführung der Protokollsicherheitstests betrieben werden kann.

[0166] Durch die vorliegende Erfindung erfolgt zudem die Bereitstellung eines mittels einer Maschine (beispielsweise eines Computers) lesbaren Speichermediums, welches die Befehle zur Ausführung des Verfahrens zur Durchführung von Sicherheitstests von Protokollimplementierungen gemäß der vorliegenden Erfindung durch eine Maschine speichert.

[0167] Durch die vorliegende Erfindung erfolgt zudem die Bereitstellung eines Computerprogramms, welches computerlesbare Befehle zur Ausführung des Verfahrens zur Durchführung von Sicherheitstests von Protokollimplementierungen gemäß der vorliegenden Erfindung ausführt.

[0168] Konkret wird vorgesehen, dass die Bereitstellung von einem System beziehungsweise einer Vorrichtung erfolgt, welches beziehungsweise welche mit einem Speichermedium ausgestattet ist, wobei in dem betreffenden Speichermedium die Softwareprogrammcodes für die praktische Umsetzung der Funktionen eines beliebigen der vorstehend aufgeführten praktischen Ausführungsbeispiele gespeichert sind, so dass bewirkt wird, dass durch den Computer (beziehungsweise CPU beziehungsweise MPU) des betreffenden Systems oder der betreffenden Vorrichtung das Auslesen und Ausführen der in dem Speichermedium gespeicherten Programmcodes erfolgt.

[0169] In diesem Fall können die aus dem Speichermedium ausgelesenen Programmcodes an sich die Funktionen eines beliebigen der vorstehend aufgeführten praktischen Ausführungsbeispiele praktisch umsetzen. Somit stellt das Speichermedium für die Programmcodes und die Speicherprogrammcodes einen Teil der vorliegenden Erfindung dar.

[0170] Praktische Ausführungsbeispiele der für die Bereitstellung von Programmcodes verwendeten Speichermedien sind flexible Disketten, feste Disketten, Magnetplatten, CDs (beispielsweise CD-ROM, CD-R, CD-RW, DVD-ROM, DVD-RAM, DVD-RW, DVD+RW), Magnetbänder, verlustgesicherte Speicherkarten und ROM. Optional kann vorgesehen werden, dass über das Internet von einem Server Programmcodes heruntergeladen werden.

[0171] Es ist außerdem zu beachten, dass nicht lediglich durch die Ausführung von durch Computer ausgelesenen Programmcodes, sondern auch durch auf Programmcodes basierende Befehle bewirkt wird, dass ein auf dem Computer aktives Bedienungssystem usw. die tatsächliche Bedienung teilweise beziehungsweise vollständig vornehmen kann und auf diese Weise die Funktionen eines beliebigen der vorstehend aufgeführten praktischen Ausführungsbeispiele praktisch umsetzen kann.

[0172] Bei den praktischen Ausführungsbeispielen der vorliegenden Erfindung wird vorgesehen, dass einem auf Protokollspezifikationen basierenden, formalisierten Verfahren die hauptsächliche Funktion bei den Protokollsicherheitstest zukommt, wobei die Protokollspezifikationen verwendet werden können, um die automatische Erzeugung der für den automatischen Test der im Test befindlichen Implementierung IUT verwendeten

Gegenbeispiele anzuleiten. Auf diese Weise kann das Arbeitsaufkommen für die Protokollsicherheitstests mit Gegenbeispielen signifikant vermindert werden und es erfolgt gleichzeitig eine signifikante Steigerung der Effizienz der Protokollsicherheitstests. Zudem kann der durch die praktischen Ausführungsbeispiele der vorliegenden Erfindung vorgeschlagene technische Planungsentwurf einfach umgesetzt werden, ist unabhängig von der konkreten Protokollimplementierung und bietet eine hohe Praktikabilität.

[0173] Bei den vorstehend aufgeführten Erläuterungen handelt es sich lediglich um bevorzugte praktische Ausführungsbeispiele der vorliegenden Erfindung, ohne eine Beschränkung der vorliegenden Erfindung darzustellen. Sämtliche ohne Abweichung von der wesentlichen technischen Lehre der vorliegenden Erfindung vorgenommenen Abänderungen, Äquivalente, Ersetzungen oder Verbesserungen fallen sämtlich in den Schutzbereich der vorliegenden Erfindung.

Patentansprüche

1. Verfahren zur Durchführung von Sicherheitstests hinsichtlich von Protokollimplementierungen, wobei das betreffende Verfahren Folgendes beinhaltet:

- Abbilden der Protokollspezifikationen, welche die Grundlage der vorstehend bezeichneten Protokollimplementierung darstellen, zu einem ersten Protokollstatusmodell;
- Einführen eines nicht ordnungsgemäßen Status zur Erzeugung eines zweiten Protokollstatusmodells in das vorstehend bezeichnete erste Protokollstatusmodell, wobei der vorstehend bezeichnete nicht ordnungsgemäße Status in der vorstehend bezeichneten Protokollimplementierung existierenden Sicherheitsstörungen entspricht;
- unter Verwendung kategorischer Festlegungen, Repräsentieren von Testzielen des vorstehend bezeichneten ersten Protokollstatusmodells und/oder des vorstehend bezeichneten zweiten Protokollstatusmodells;
- unter Verwendung syntaktischer Beschränkungen, Repräsentieren der syntaktischen Bestimmungen für ordnungsgemäße Eingabemeldungen von vorstehend bezeichnetem ersten Protokollstatusmodell und/oder vorstehend bezeichnetem zweiten Protokollstatusmodell;
- hinsichtlich des vorstehend bezeichneten zweiten Protokollstatusmodells, Ausführen der Verhaltensmutationsoperation, um ein mutiertes Protokollstatusmodell zu erzeugen;
- entsprechend den vorstehend bezeichneten kategorischen Festlegungen, Durchführen des Modelltests hinsichtlich des vorstehend bezeichneten mutierten Protokollstatusmodells und Erzeugen eines den vorstehend bezeichneten kategorischen Festlegungen widersprechenden Gegenbeispiels oder einer Mehrzahl von solchen Gegenbeispielen, wobei die vorstehend bezeichneten Gegenbeispiele zu Testabfolgen abgebildet werden;
- hinsichtlich der vorstehend bezeichneten Testabfolgen, Ausführen der Syntaxmutationsoperation, um mutierte Meldungen enthaltende Testabfolgen zu erzeugen, wobei jede Testabfolge eine mutierte Testabfolge erzeugt, welche eine mutierte Meldung oder eine Mehrzahl von solchen mutierten Meldungen enthält;
- entsprechend den vorstehend bezeichneten syntaktischen Beschränkungen, Durchführen des Syntaxtests hinsichtlich der in den vorstehend bezeichneten mutierten Testabfolgen befindlichen mutierten Meldungen, um in den vorstehend bezeichneten mutierten Meldungen enthaltene und den vorstehend bezeichneten syntaktischen Beschränkungen widersprechende gültige mutierte Meldungen zu erhalten, wobei die vorstehend bezeichnete gültige mutierte Meldungen enthaltenden vorstehend bezeichneten mutierten Testabfolgen als Testanwendungsbeispiele verwendet werden; und
- unter Verwendung der vorstehend bezeichneten Testanwendungsbeispiele, Durchführen von Fuzz-Testing hinsichtlich der Protokollimplementierung, um zu testen, ob in der vorstehend bezeichneten Protokollimplementierung einem vorstehend bezeichneten Testanwendungsbeispiel oder einer Mehrzahl von solchen Testanwendungsbeispielen entsprechende Sicherheitslücken existieren.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass es sich bei dem vorstehend bezeichneten ersten Protokollstatusmodell um einen parameterisierten erweiterten endlichen Zustandsautomaten PEFSM beziehungsweise um ein Input Output Labeled Transition System IOLTS handelt.

3. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass die vorstehend bezeichnete Verwendung von kategorischen Festlegungen zur Repräsentierung von Testzielen des vorstehend bezeichneten ersten Protokollstatusmodells und/oder des vorstehend bezeichneten zweiten Protokollstatusmodells Folgendes beinhaltet:

- basierend auf den gegebenen Testzielen Definieren von kategorischen Festlegungen; und/oder
- entsprechend von aus dem Bereich der Sicherheitstests stammendem Wissen, vorab Definieren von kategorischen Festlegungen; und/oder
- aus dem vorstehend bezeichneten ersten Protokollstatusmodell, Reflexieren von kategorischen Festlegungen.

4. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass die vorstehend bezeichnete Verwendung syntaktischer Beschränkungen Folgendes beinhaltet:
entsprechend den Protokollspezifikationen, Definieren von syntaktischen Beschränkungen; und/oder aus den ordnungsgemäßen Eingabemeldungen des vorstehend bezeichneten ersten Protokollstatusmodells, Reflexieren von syntaktischen Beschränkungen.
5. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass die vorstehend bezeichnete Ausführung der Verhaltensmutationsoperation hinsichtlich des vorstehend bezeichneten zweiten Protokollstatusmodells Folgendes beinhaltet:
unter Verwendung von Verhaltensmutationsoperatoren, Ausführen der Verhaltensmutationsoperation hinsichtlich der Übertragung in dem vorstehend bezeichneten zweiten Protokollstatusmodell sowie hinsichtlich der logischen Operationen bei der Übertragung.
6. Verfahren nach Anspruch 5, dadurch gekennzeichnet, dass die vorstehend bezeichnete Ausführung der Verhaltensmutationsoperation hinsichtlich der Übertragung in dem vorstehend bezeichneten zweiten Protokollstatusmodell Folgendes beinhaltet:
unter Verwendung von Statusfallenoperator und/oder Eingabeveränderungsoperator, Durchführen der Verhaltensmutationsoperation hinsichtlich der vorstehend bezeichneten Übertragung, wobei der vorstehend bezeichnete Statusfallenoperator verwendet wird, um den Status nach erfolgter Übertragung in einen nicht ordnungsgemäßen Status zu verändern, während der vorstehend bezeichnete Eingabeveränderungsoperator verwendet wird, um die Eingabemeldungen der Übertragung in andere Meldungen zu verändern.
7. Verfahren nach Anspruch 5, dadurch gekennzeichnet, dass die vorstehend bezeichnete Ausführung der Verhaltensmutationsoperation hinsichtlich der logischen Operationen bei der Übertragung in dem vorstehend bezeichneten zweiten Protokollstatusmodell Folgendes beinhaltet:
unter Verwendung von einem beliebigen der nachstehend aufgeführten Operatoren beziehungsweise einer beliebigen Kombination der nachstehend aufgeführten Operatoren: Expression Negation Operator, Expression Increase Operator, Expression Decrease Operator, stuck-at-0-Operator, stuck-at-1-Operator, Operand Negation Operator, Operand Reference Operator, Range Exploration Operator und Case Injection Operator, Ausführen der Verhaltensmutationsoperation hinsichtlich der vorstehend bezeichneten logischen Operationen, wobei:
der vorstehend bezeichnete Expression Negation Operator verwendet wird, um den Expressionswert in der logischen Operation umzukehren;
der vorstehend bezeichnete Expression Increase Operator verwendet wird, um den Expressionswert in der logischen Operation um n zu vermehren;
der vorstehend bezeichnete Expression Decrease Operator verwendet wird, um den Expressionswert in der logischen Operation um n zu vermindern;
der vorstehend bezeichnete stuck-at-0-Operator verwendet wird, um den Expressionswert in der logischen Operation auf 0 zu verändern,
der vorstehend bezeichnete stuck-at-1-Operator verwendet wird, um den Expressionswert in der logischen Operation auf 1 zu verändern;
der vorstehend bezeichnete Operand Negation Operator verwendet wird, um die verschiedenen Operanden in der Expression nach Stellen umzukehren;
der vorstehend bezeichnete Operand Reference Operator verwendet wird, um die Operanden nacheinander mit ihrem nachfolgenden Operanden auszutauschen und den letzten Operanden mit dem ersten Operanden auszutauschen;
der vorstehend bezeichnete Range Exploration Operator verwendet wird, um den Wert von jedem Operanden in der Expression nach Stellen umzukehren und die betreffende Zuweisungsanweisung mit der Kennzeichnung "Ersetzung" zu versehen;
und der vorstehend bezeichnete Case Injection Operator verwendet wird, um in der logischen Operation befindliche, in ihrer Länge variable Felder um n Byte zu vermehren oder zu vermindern und die betreffende Zuweisungsanweisung mit der Kennzeichnung "Injektion" zu versehen,
wobei es sich bei n um eine positive ganze Zahl handelt.
8. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass jedes der für den Modelltest erzeugten Gegenbeispiele zu einer Testabfolge abgebildet wird, wobei das betreffende Gegenbeispiel einem in dem vorstehend bezeichneten mutierten Protokollstatusmodell vorhandenen Übertragungsweg entspricht, welcher von einer Übertragung oder einer Mehrzahl von solchen Übertragungen gebildet wird, wobei auf den vorstehend bezeichneten Übertragungswegen eine Eingabemeldung oder eine Mehrzahl von solchen Eingabemeldungen sowie eine diesen entsprechende Ausgabemeldung oder eine Mehrzahl von solchen Ausgabemeldungen existiert;

wobei das vorstehend bezeichnete Abbilden des vorstehend bezeichneten Gegenbeispiels zu einer Testabfolge Folgendes beinhaltet: Abbilden des vorstehend bezeichneten Gegenbeispiels zu einer auf dem vorstehend bezeichneten Übertragungsweg existierenden Eingabemeldung oder einer Mehrzahl von solchen Eingabemeldungen sowie zu einer diesen entsprechenden Ausgabemeldung oder einer Mehrzahl von solchen Ausgabemeldungen, wobei die Abfolge der erhaltenen Meldungen die Abfolge der vorstehend bezeichneten Testabfolge ist, und Kennzeichnen einer beliebigen, den kategorischen Festlegungen widersprechenden mutierten Übertragungen entsprechenden Meldung sowie der in den vorstehend bezeichneten Meldungen durch Modelltest ermittelten Felder.

9. Verfahren nach Anspruch 8, dadurch gekennzeichnet, dass die vorstehend bezeichnete Kennzeichnung einer beliebigen, den kategorischen Festlegungen widersprechenden mutierten Übertragungen entsprechenden Meldung sowie der in den vorstehend bezeichneten Meldungen durch Modelltest ermittelten Felder Folgendes beinhaltet:

falls der Modelltest die Identifizierung der Mutation eines gewissen Feldes in der vorstehend bezeichneten Meldung mit möglicher Herbeiführung des betreffenden Gegenbeispiels ermöglicht und das vorstehend bezeichnete Feld über keine andere Kennzeichnung verfügt, Kennzeichnen des vorstehend bezeichneten Feldes mit "Fuzz-Testing", während anderenfalls die gesamte Meldung mit der Kennzeichnung "Fuzz-Testing" versehen wird;

falls der Modelltest die Identifizierung der Kennzeichnung einer einem gewissen Feld in der vorstehend bezeichneten Meldung entsprechenden logischen Operation als "Ersetzung" ergibt, Kennzeichnen für das vorstehend bezeichnete in der Meldung enthaltene Feld mit "Ersetzung"; und

falls der Modelltest die Identifizierung der Kennzeichnung einer einem gewissen Feld in der vorstehend bezeichneten Meldung entsprechenden logischen Operation als "Injektion" ergibt, Kennzeichnen für das vorstehend bezeichnete in der Meldung enthaltene Feld mit "Injektion".

10. Verfahren nach Anspruch 9, dadurch gekennzeichnet, dass bei der vorstehend bezeichneten Ausführung der Syntaxmutationsoperation hinsichtlich der vorstehend bezeichneten Testabfolge bezüglich von in der Testabfolge enthaltenen und mit der Kennzeichnung "Fuzz-Testing" versehenen Meldungen beziehungsweise bezüglich von mit der Kennzeichnung "Fuzz-Testing" versehene Felder enthaltenden Meldungen unter Verwendung des Syntaxmutationsoperators die Ausführung der Syntaxmutationsoperation erfolgt.

11. Verfahren nach Anspruch 10, dadurch gekennzeichnet, dass der vorstehend bezeichnete Syntaxmutationsoperator einen beliebigen der nachstehend aufgeführten Syntaxmutationsoperatoren beziehungsweise eine beliebige Kombination der nachstehend aufgeführten Syntaxmutationsoperatoren beinhaltet:

Feldlöschungsoperator, welcher verwendet wird, um mit der Kennzeichnung "Fuzz-Testing" versehene Felder in den Meldungen zu löschen;

Feldwiederholungsoperator, welcher verwendet wird, um mit der Kennzeichnung "Fuzz-Testing" versehene Felder in den Meldungen zu mehreren Kopien zu kopieren;

Feldaustauschoperator, welcher verwendet wird, um die Position von mit der Kennzeichnung "Fuzz-Testing" versehenen Feldern mit der Position des nachfolgenden Feldes auszutauschen;

Breitenveränderungsoperator, welcher verwendet wird, um den Inhalt von mit der Kennzeichnung "Fuzz-Testing" versehenen Feldern um n Byte zu vermehren beziehungsweise zu vermindern, wobei es sich bei n um eine positive ganze Zahl handelt;

Codeveränderungsoperator, welcher verwendet wird, um das Codeformat von mit der Kennzeichnung "Fuzz-Testing" versehenen Feldern zu einem anderen Format zu verändern; sowie

Wertnegierungsoperator, welcher verwendet wird, um den Wert von mit der Kennzeichnung "Fuzz-Testing" versehenen Feldern umzukehren beziehungsweise nach Stellen umzukehren.

12. Verfahren nach Anspruch 9, dadurch gekennzeichnet, dass bei der vorstehend bezeichneten Ausführung der Syntaxmutationsoperation hinsichtlich der vorstehend bezeichneten Testabfolge bezüglich von in Testanwendungsbeispielen vorhandenen, mit der Kennzeichnung "Ersetzung" versehene Felder enthaltenden Meldungen unter Verwendung vorab definierter Testwerte beziehungsweise unter Verwendung von den kategorischen Festlegungen widersprechenden nicht ordnungsgemäßen Werten die Ersetzung der in der betreffenden Meldung mit der Kennzeichnung "Ersetzung" versehenen Felder erfolgt und die entsprechenden mutierten Meldungen erhalten werden;

wobei hinsichtlich von in Testanwendungsbeispielen vorhandenen, mit der Kennzeichnung "Injektion" versehene Felder enthaltenden Meldungen unter Verwendung vorab definierter Testwerte beliebiger Länge beziehungsweise unter Verwendung von den kategorischen Festlegungen widersprechenden nicht ordnungsgemäßen Werten die Injektion in die in der betreffenden Meldung mit der Kennzeichnung "Injektion" versehenen Felder erfolgt und die entsprechenden mutierten Meldungen erhalten werden.

13. Verfahren nach Anspruch 9, dadurch gekennzeichnet, dass das betreffende Verfahren weiterhin Folgendes beinhaltet: unmittelbares Verwenden von Testabfolgen, welche lediglich mit der Kennzeichnung "Ersetzung" versehene beziehungsweise mit der Kennzeichnung "Injektion" versehene Felder enthalten, als Testanwendungsbeispiele zur Durchführung von Fuzz-Testing.

14. Verfahren nach Anspruch 12 oder 13, dadurch gekennzeichnet, dass die vorstehend bezeichnete Verwendung der vorstehend bezeichneten Testanwendungsbeispiele zur Durchführung von Fuzz-Testing hinsichtlich der Protokollimplementierung Folgendes beinhaltet:

Versenden der in den betreffenden Testanwendungsbeispielen enthaltenen jeweiligen Eingabemeldungen nacheinander an die vorstehend bezeichnete Protokollimplementierung; und hinsichtlich der von der vorstehend bezeichneten Protokollimplementierung empfangenen Reaktionsmeldungen wird vorgesehen, dass wenn irgendeine der empfangenen Reaktionsmeldungen mit den in den betreffenden Testanwendungsbeispielen enthaltenen Ausgabemeldungen nicht übereinstimmt, beziehungsweise sich die vorstehend bezeichnete Protokollimplementierung in einem nicht ordnungsgemäßen Status befindet, somit festgestellt wird, dass bei der vorstehend bezeichneten Protokollimplementierung den entsprechenden Eingabemeldungen entsprechende Sicherheitslücken existieren.

15. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass das betreffende Verfahren weiterhin beinhaltet, dass vor der Durchführung der Syntaxmutationsoperation eine Optimierungsbehandlung der vorstehend bezeichneten einen Testabfolge oder einer Mehrzahl von solchen Testabfolgen durchgeführt wird, um redundante Testabfolgen zu entfernen.

16. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass das vorstehend bezeichnete erste Protokollstatusmodell den betreffenden Protokollstatus und die Statusübertragung beschreibt.

17. Vorrichtung zur Durchführung von Sicherheitstests hinsichtlich der Protokollimplementierung, dadurch gekennzeichnet, dass die betreffende Vorrichtung Folgendes beinhaltet:

ein erstes Modul (**601**), welches verwendet wird, um während der Durchführung des Sicherheitstests hinsichtlich der vorstehend bezeichneten Protokollimplementierung das Abbilden der vorstehend bezeichneten Protokollimplementierung zugrunde liegenden Protokollspezifikationen zu einem ersten Protokollstatusmodell vorzunehmen;

ein zweites Modul (**602**), welches verwendet wird, um in das von dem vorstehend bezeichneten ersten Modul (**601**) erhaltene vorstehend bezeichnete erste Protokollstatusmodell einen nicht ordnungsgemäßen Status einzuführen und das zweite Protokollstatusmodell zu erzeugen, wobei der vorstehend bezeichnete nicht ordnungsgemäße Status bei der vorstehend bezeichneten Protokollimplementierung existierenden Sicherheitsstörungen entspricht;

ein drittes Modul (**603**), welches verwendet wird, um unter Verwendung kategorischer Festlegungen die Repräsentierung von Testzielen des von dem vorstehend bezeichneten zweiten Modul (**602**) stammenden vorstehend bezeichneten zweiten Protokollstatusmodells und/oder des von dem vorstehend bezeichneten ersten Modul (**601**) stammenden vorstehend bezeichneten ersten Protokollstatusmodells vorzunehmen, wobei unter Verwendung syntaktischer Beschränkungen die Repräsentierung der syntaktischen Bestimmungen für ordnungsgemäße Eingabemeldungen des vorstehend bezeichneten zweiten Protokollstatusmodells und/oder des vorstehend bezeichneten ersten Protokollstatusmodells erfolgt;

ein viertes Modul (**604**), welches verwendet wird, um hinsichtlich des von dem vorstehend bezeichneten zweiten Modul (**602**) stammenden vorstehend bezeichneten zweiten Protokollstatusmodells die Verhaltensmutationsoperation vorzunehmen und auf diese Weise das mutierte Protokollstatusmodell zu erzeugen;

ein fünftes Modul (**605**), welches verwendet wird, um unter Verwendung der von dem vorstehend bezeichneten dritten Modul (**603**) erhaltenen kategorischen Festlegungen hinsichtlich des von dem vorstehend bezeichneten vierten Modul (**604**) erzeugten vorstehend bezeichneten mutierten Protokollstatusmodells den Modelltest durchzuführen und eine Testabfolge oder eine Mehrzahl von solchen Testabfolgen zu erzeugen, wobei die vorstehend bezeichnete eine Testabfolge oder eine Mehrzahl von solchen Testabfolgen durch Abbilden eines in dem vorstehend bezeichneten mutierten Protokollstatusmodell enthaltenen, den vorstehend bezeichneten kategorischen Festlegungen widersprechenden Gegenbeispiels oder einer Mehrzahl von solchen Gegenbeispielen erhalten wird;

ein sechstes Modul (**606**), welches verwendet wird, um hinsichtlich der von dem vorstehend bezeichneten fünften Modul (**605**) stammenden Testabfolge die Syntaxmutationsoperation vorzunehmen und die mutierte Meldungen enthaltende mutierte Testabfolge zu erzeugen, wobei eine Testabfolge eine Mutationen enthaltende mutierte Testabfolge oder eine Mehrzahl von solchen Mutationen enthaltenden mutierten Testabfolgen erzeugt;

ein siebtes Modul (**607**), welches verwendet wird, um unter Verwendung der von dem vorstehend bezeichneten dritten Modul (**603**) erhaltenen syntaktischen Beschränkungen hinsichtlich der von dem vorstehend bezeichneten sechsten Modul (**606**) erzeugten mutierten Testabfolge den Syntaxtest durchzuführen und die in den vorstehend bezeichneten mutierten Meldungen enthaltenen, den vorstehend bezeichneten syntaktischen Beschränkungen widersprechenden gültigen mutierten Meldungen zu erhalten, wobei die vorstehend bezeichnete gültige mutierte Meldungen enthaltenden mutierten Testabfolgen als Testanwendungsbeispiele verwendet werden; sowie

ein achtes Modul (**608**), welches verwendet wird, um unter Verwendung der von vorstehend bezeichnetem siebten Modul (**607**) erhaltenen Testanwendungsbeispiele hinsichtlich der Protokollimplementierung die Durchführung von Fuzz-Testing vorzunehmen, um zu testen, ob bei der vorstehend bezeichneten Protokollimplementierung einem Testanwendungsbeispiel oder einer Mehrzahl von solchen Testanwendungsbeispielen entsprechende Sicherheitslücken existieren.

18. Vorrichtung nach Anspruch 17, dadurch gekennzeichnet, dass das vorstehend bezeichnete dritte Modul (**603**) weiterhin verwendet wird, um die Optimierung der vorstehend bezeichneten kategorischen Festlegungen vorzunehmen und um redundante kategorische Festlegungen zu entfernen, beziehungsweise dass das vorstehend bezeichnete fünfte Modul (**605**) weiterhin verwendet wird, um die Optimierung des vorstehend bezeichneten einen Testanwendungsbeispiels oder einer Mehrzahl von solchen Testanwendungsbeispielen vorzunehmen, um redundante Testanwendungsbeispiele zu entfernen.

19. Vorrichtung nach Anspruch 17 oder 18, dadurch gekennzeichnet, dass das vorstehend bezeichnete fünfte Modul (**605**) weiterhin verwendet wird, um unmittelbar die Verwendung von Testabfolgen, welche lediglich mit der Kennzeichnung "Ersetzung" versehene beziehungsweise mit der Kennzeichnung "Injektion" versehene Felder enthalten, als Testanwendungsbeispiele vorzunehmen und an das vorstehend bezeichnete achte Modul (**608**) zur Durchführung von Fuzz-Testing auszugeben.

20. Vorrichtung nach Anspruch 17, dadurch gekennzeichnet, dass das vorstehend bezeichnete erste Modul (**601**) so ausgestaltet wird, dass Abbilden der vorstehend bezeichneten Protokollspezifikationen zu dem betreffenden ersten Protokollstatusmodell erfolgt, welches den betreffenden Protokollstatus und die Statusübertragung beschreibt.

21. Computerlesbares Medium, in welchem computerlesbare Codes zur Ausführung des Verfahrens nach einem beliebigen der Ansprüche 1–16 gespeichert sind.

22. Computerprogramm, welches computerlesbare Codes zur Ausführung des Verfahrens nach einem beliebigen der Ansprüche 1–16 beinhaltet.

Es folgen 4 Blatt Zeichnungen

Anhängende Zeichnungen

FIG 1

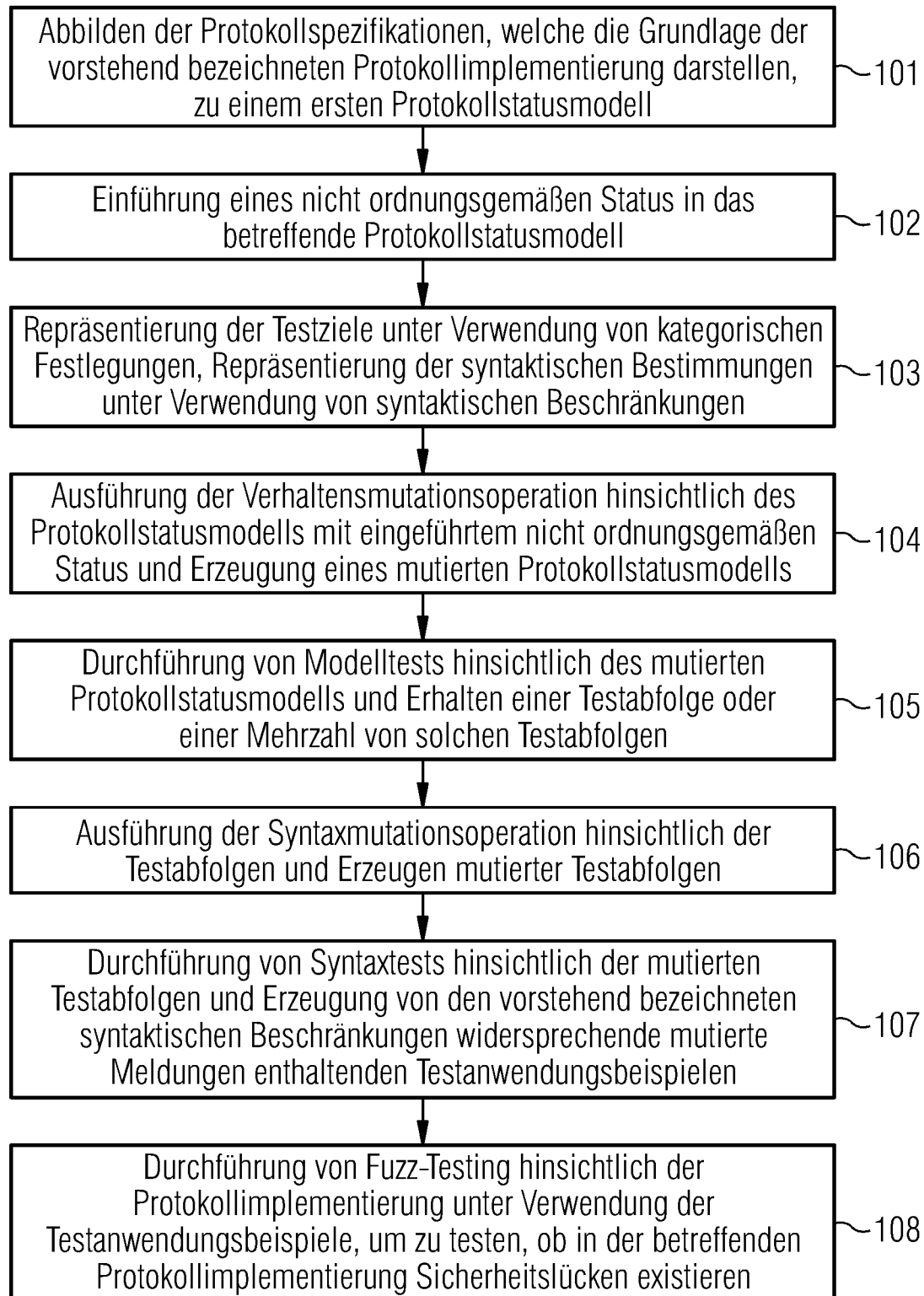


FIG 2

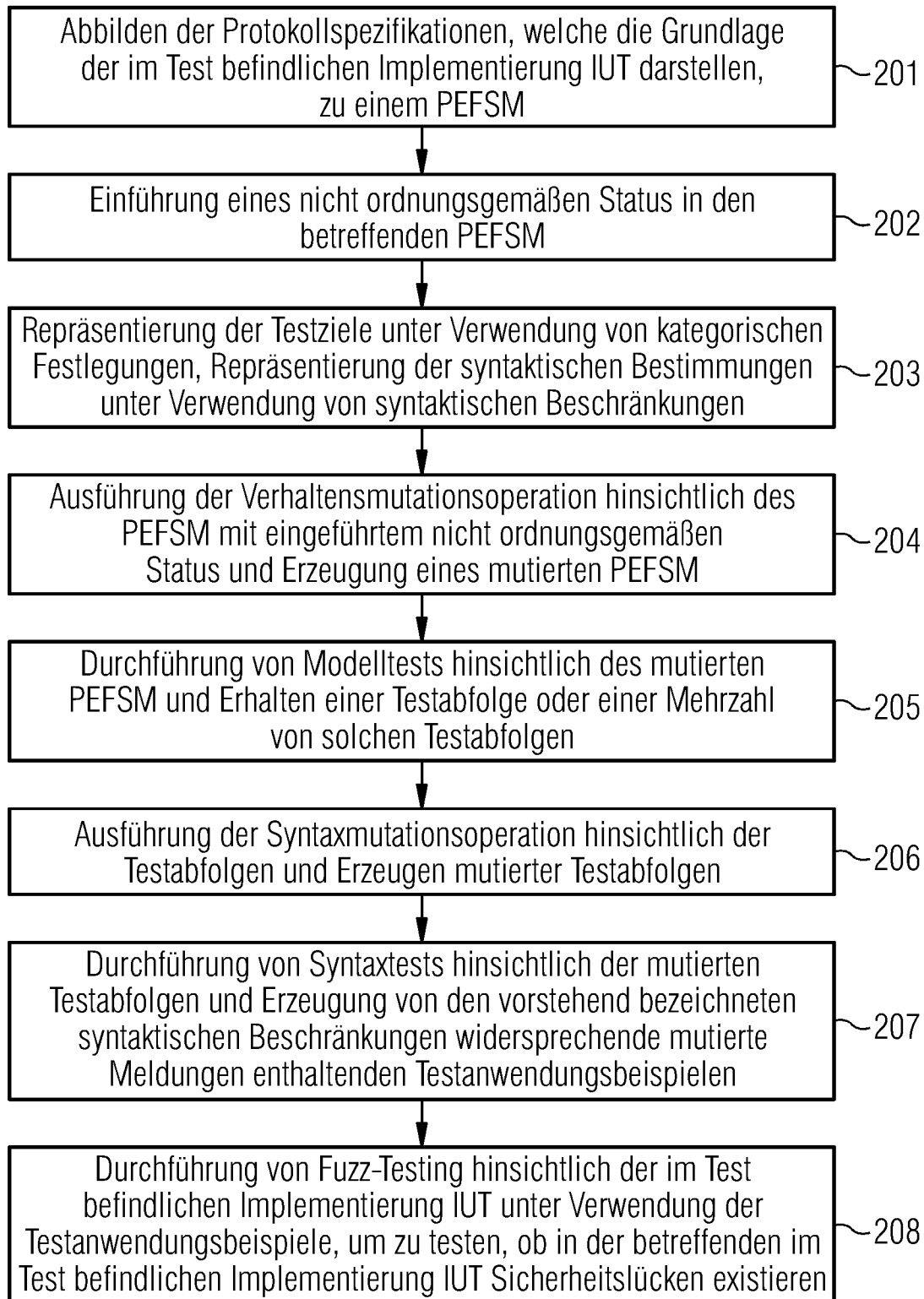


FIG 3

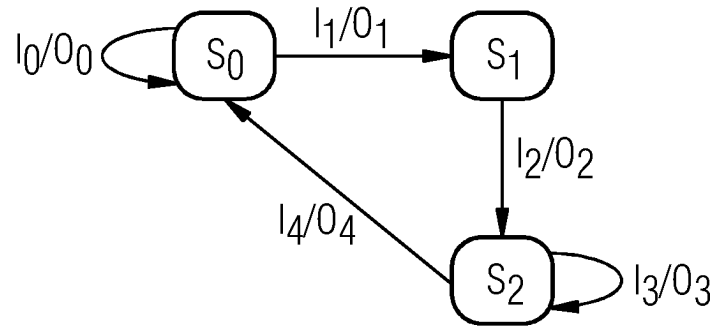


FIG 4

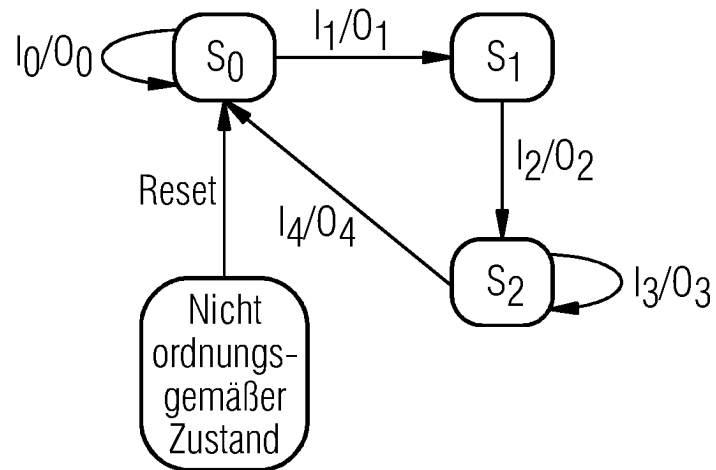


FIG 5

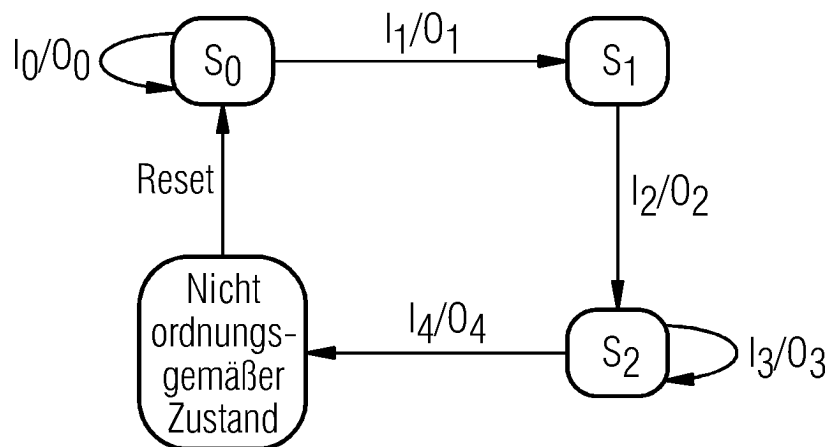


FIG 6

