Zur Anwendbarkeit von TESDL für die automatische Generierung von TTCN Testfällen aus SDL Spezifikationen -

Eine auf dem Inres Protokoll basierende Fallstudie

Jens Grabowski Hansruedi Scheurer

Andreas Spichiger Stefan Suter

IAM 93 - 009

April 1993

Zur Anwendbarkeit von TESDL für die automatische Generierung von TTCN Testfällen aus SDL Spezifikationen -

Eine auf dem Inres Protokoll basierende Fallstudie

Jens Grabowski Hansruedi Scheurer

Andreas Spichiger Stefan Suter

Überblick: Die vorliegende Fallstudie beschäftigt sich mit der automatischen Generierung von Testfällen aus SDL Spezifikationen. Basierend auf dem Inres Beispiel wird gezeigt, wie sich mit Hilfe des Software Programms TESDL Testfälle generieren lassen. TESDL generiert TTCN Verhaltensbäume für das beobachtbare (externe) Verhalten einer SDL Spezifikation durch dessen Simulation. Dabei wird jeder mögliche Simulationsweg bis zu einer vorher definierten Baumtiefe berücksichtigt. Unterschiedliche Simulationswege können jedoch das gleiche beobachtbare Verhalten haben. Bei einem Black Box Test sind unterschiedliche Systemabläufe mit dem gleichen beobachtbaren Verhalten deshalb nicht unterscheidbar. Bezogen auf das Testen können deshalb die mit TESDL generierten Verhaltensbäume Redundanz beinhalten.

In dieser Fallstudie wird beschrieben, wie Testfälle für die unterschiedlichen Kommunikationsphasen des Inres Protokolls mit TESDL generiert werden können. Ferner werden die Einflüsse von unterschiedlichen Testkonfigurationen auf die Testfallgenerierung und die Resultate unserer Experimente mit TESDL beschrieben. Desweiteren beinhaltet diese Fallstudie eine Beschreibung, wie die erwähnte Redundanz aus den generierten Verhaltensbäumen entfernt werden, und einen Vergleich zwischen den ursprünglichen und den redundanzfreien TTCN Verhaltensbäumen.

Klassifikation nach Computing Reviews: C.2.0 [Computer-Communication Networks]: General; C.2.0 [Computer-Communication Networks]: Network Protocols; D.2.5 [Software Engineering]: Testing and Debugging

On the Applicability of TESDL for the Automatic Generation of TTCN Test Cases from SDL Specifications -

A case study based on the Inres protocol

Jens Grabowski Hansruedi Scheurer

Andreas Spichiger Stefan Suter

Abstract: This case study deals with the automatic generation of test cases from SDL specifications. Based on the Inres example it explains how the tool TESDL can be utilised for generating test cases. TESDL generates a TTCN behaviour tree of the observable behaviour of an SDL specification by simulation, during which the observable behaviour of every possible simulation path up to a certain depth is reported. Unfortunately, different simulation paths may show the same observable behaviour. A black box test cannot distinguish between different system executions with the same observable behaviour and therefore, TTCN behaviour trees generated by TESDL may include redundancy.

In this case study it is shown how test cases for the different communication phases of the Inres protocol can be generated using TESDL. Furthermore, the influences of different test configurations on the test case generation and the results of our TESDL experiments are described. Additionally we have eliminated the redundancy of the TESDL outputs. How this is done, together with a comparison of the original and the redundance free TTCN behaviour trees can be found in the last section of this case study.

CR Categories and Subject Descriptors: C.2.0 [Computer-Communication Networks]: General; C.2.0 [Computer-Communication Networks]: Network Protocols; D.2.5 [Software Engineering]: Testing and Debugging

0 Inhalt

1 Einleitung

Aufgrund der zunehmenden Standardisierung der Kommunikationsprotokolle von Computersystemen in heterogenen Netzwerken wird Konformitätstesten zunehmend wichtig. Mit einem Konformitätstest wird die Funktionalität einer Protokollimplementation getestet. Ist ein Protokoll mit einer formalen Sprache (z.B. LOTOS, Estelle, SDL) spezifiziert, so können aus der Protokollspezifikation Testfälle für Konformitätstests automatisch generiert werden.

Das ISO Referenz Modell [ISO 7498] besagt, dass das externe Verhalten eines offenen Systems das Standardverhalten ist. Konformitätstesten ist also ein Black Box Testen. Die ISO schlägt für die Darstellung von Testfällen für Konformitätstests TTCN (Tree and Tabular Combined Notation) [ISO 9646-3] vor. In dieser Arbeit werden TTCN Verhaltensbäume für ein mit SDL (für SDL vgl. [BrHoSa 91], [Hog 89], [Z100]) spezifiziertes Protokoll generiert.

1.1 TESDL

Das in dieser Fallstudie zur Testdatengenerierung verwendete Werkzeug TESDL ist im Rahmen des Forschungsprojektes PROSPECT entwickelt worden und auf PCs ablauffähig (vgl. [Brö 89], [BrDoHo 90], [BröHog 89], [Hog 88], [PROS 90]). TESDL simuliert eine SDL Spezifikation und generiert TTCN Verhaltensbäume[1] für das externe, d.h. nach aussen beobachtbare, Verhalten der simulierten Spezifikation. Bei der Testdatengenerierung berücksichtigt TESDL alle aus dem Initialzustand der Spezifikation möglichen Simulationspfade. Als Abbruchkriterium für die Testdatengenerierung muss die Tiefe des zu generierenden TTCN Verhaltensbaum angegeben werden. TESDL erzeugt während der Simulation automatisch eventuell notwendige Eingabedaten für die simulierte Spezifikation.

1.2 Inres

Das verwendete Protokollbeispiel ist das Inres Protokoll (Abb.1). Eine vollständige Beschreibung des Inres Systems ist in [Hog 92] zu finden. Das Inres Protokoll ist eine asymmetrische Version des Abracadabra Protokolls [ISO 10167]. Nur eine Seite des Protokolls, der sog. Initiator, kann eine Verbindung aufbauen und Daten senden. Die andere Seite, der Empfänger, nimmt die Anforderungen eines Verbindungsaufbaus und die Daten entgegen. Er muss die empfangenen Daten bestätigen und kann die bestehende Verbindung jederzeit abbrechen. Der Initiator wiederholt unbestätigte Verbindungsaufbauwünsche und unbestätigte Daten maximal dreimal. Wenn keine Bestätigung erfolgt, wird ein Verbindungsabbau eingeleitet. Das Protokoll

[1] TESDL generiert keine TTCN Testfälle, sondern schreibt das beobachtbare Verhalten der simulierten SDL Spezifikation in der TTCN typischen Baumdarstellung auf. Es sei jedoch darauf hingewiesen, dass die mit TESDL generierten Bäume im allgemeinen nicht mit der TTCN Semantik konform sind. Der Einfachheit halber werden die mit TESDL erzeugten Verhaltensbäume als *Testfälle* oder *Testdaten* bezeichnet. Entsprechend wird die Generierung der TTCN Verhaltensbäume auch *Testdatengenerierung* genannt.

benutzt einen unsicheren Medium Service, um seinen Service zu realisieren. Auf Grund der Komplexität der mit TESDL generierbaren Testdaten wurde in dieser Fallstudie ein sicheres Medium modelliert.

Das Inres Protokoll kann keiner speziellen OSI Schicht zugeordnet werden, obwohl es viele OSI Konzepte enthält. Es hat sich auf Grund seiner Übersichtlichkeit zu einer Art Standardbeispiel im akademischen Bereich entwickelt.

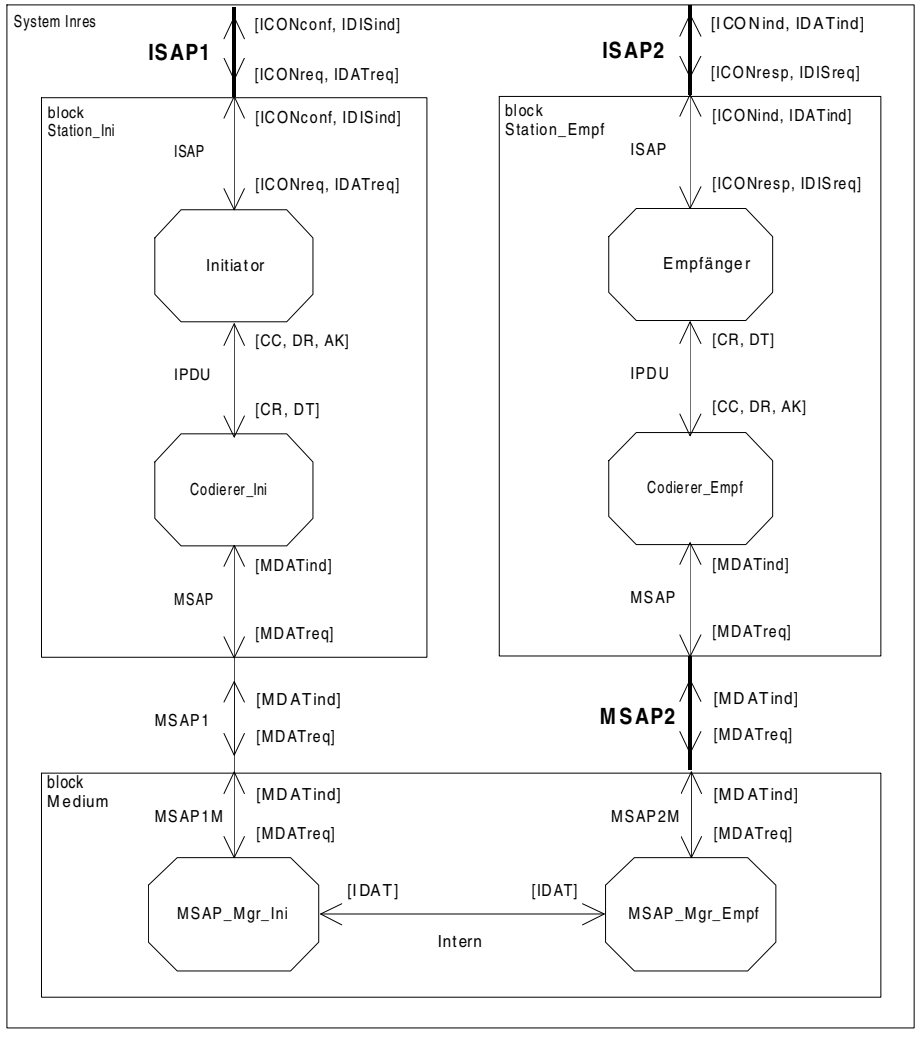


Abbildung 1: SDL Blockdiagramm für das Inres Protokoll

2 Testaufbau

Dieser Abschnitt beschreibt den Testaufbau, mit dem durch TESDL Testdaten generiert werden. Zunächst werden drei verschiedene Beobachtungspunkte, sogenannte PCOs (Points of Control and Observation), im SDL Blockdiagramm des Inres Protokolls ausgewählt. Durch die Kombi-

nation der PCOs und durch zwei verschiedene SDL Spezifikationen werden anschliessend drei Testkonfigurationen definiert.

2.1 Gewählte Beobachtungspunkte

Um mit TESDL Testdaten aus einer SDL Spezifikation generieren zu können, müssen die PCOs festgelegt werden, von denen aus das zu testende System beobachtet und kontrolliert werden soll. Nachfolgend sollen Testdaten für den Inres Service und für den Initiator generiert werden. Hierfür werden drei PCOs benötigt. Die PCOs liegen an den SAPs (Service Access Points) ISAP1, ISAP2 und MSAP2. Im SDL Blockdiagramm des Inres Protokolls (vgl. Abb. 1) sind diese SAPs durch Channels repräsentiert.

2.2 Drei Testkonfigurationen

In diesem Abschnitt werden drei verschiedene Testkonfigurationen beschrieben. Die erste Testkonfiguration dient zum Test des Inres Services. Die anderen beiden Konfigurationen dienen zum Test des Initiator Teils des Inres Protokolls.

Testkonfiguration 1

Für den Test des durch das Inres Protokoll erbrachten Services liegen die PCOs an den SAPs ISAP1 und ISAP2. Über diese SAPs kommuniziert das Inres System mit seiner Umgebung. Beobachtbar sind diejenigen Signale, die von den Benutzern an das Inres System gesendet werden (ICONreq, IDATreq, ICONresp, IDISreq) und diejenigen, die vom Inres System an den Benutzer gesendet werden (ICONconf, IDISind, ICONind, IDATind). Die zur Testdatengenerierung verwendete SDL Spezifikation für Testkonfiguration 1 ist in Anhang A.1 zu finden.

Testkonfiguration 2

Für den Test des Initiators liegen die PCOs an den SAPs ISAP1 und MSAP2. Auf der Seite des Initiators sind alle Signale beobachtbar, die vom Benutzer an den Initiator (ICONreq, IDATreq) und die vom Initiator an dessen Benutzer (ICONconf, IDISind) gesendet werden. Auf der Seite des Empfängers sind alle Signale beobachtbar, die vom Medium zum Codierer und die vom Codierer zum Medium (MDATind(...), MDATreq(...)) gesendet werden. Bei dieser Testkonfiguration wird der Empfänger Prozess in der zur Testdatengenerierung notwendigen SDL Spezifikation nicht mitmodelliert. Dieses hat zur Folge, dass an PCO MSAP2 die in MDATreq(...) kodierten Protokolldateneinheiten (CC, DR, AK) in beliebiger Reihenfolge auftreten können. Damit die Protokolldateneinheiten für TESDL auf der Seite des Empfängers definiert sind und richtig generiert werden können, wird der Codierer des Empfängers mitmodelliert (vgl. Abb. 1). Die für Testkonfiguration 2 verwendete SDL Spezifikation ist in Anhang A.2 zu finden.

Testkonfiguration 3

Die für Testkonfiguration 3 verwendete SDL Spezifikation ist identisch mit der von Testkonfiguration 1 identisch (Anhang A.1). Die Beobachtungspunkte sind jedoch die von Testkonfiguration 2 (ISAP1, MSAP2). Im Gegensatz zu Testkonfiguration 2 wird die Reihenfolge der in MDATreq(...) kodierten Protokolldateneinheiten durch die Funktionalität des Empfängers eingeschränkt. Die Einschränkung bewirkt, dass z.B. die Sequenz <DR,AK> nicht beobachtbar ist.

3 Testfälle

Das Inres System ist verbindungsorientiert. Es lassen sich mit *Verbindungsaufbau*, *Datenübertragung* und *Verbindungsabbau* drei verschiedene Kommunikationsphasen unterscheiden. In dieser Fallstudie werden Testfälle für die drei Testkonfigurationen und für alle drei Kommunikationsphasen des Inres Systems generiert.

Bei TESDL muss als Abbruchkriterium für die Testdatengenerierung die Tiefe des zu generierenden TTCN Verhaltensbaumes angegeben werden. Durch den Timer im Initiator gibt es mehrere Möglichkeiten, eine Kommunikationsphase erfolgreich zu durchlaufen, wobei auch die Anzahl der beobachtbaren Signale unterschiedlich sein kann. Abhängig von den Beobachtungspunkten und der Kommunikationsphase gibt es jedoch eine maximale Anzahl. Für jede Kommunikationsphase ist diese maximale Anzahl als Baumtiefe bzw. Abbruchkriterium gewählt worden.

3.1 Verbindungsaufbau

Der Sequence Chart (für Sequence Charts vgl. [Z100], [GraRu 89]) in Abb. 2 zeigt den ungünstigsten Fall eines erfolgreichen Verbindungsaufbaus des Inres Systems. Es ist ersichtlich, dass an PCO ISAP1 zwei, an MSAP2 acht und an ISAP2 ebenfalls acht Signale beobachtet werden können. Für alle drei Testkonfigurationen ergibt sich eine Baumtiefe von zehn für den Verbindungsaufbau.

3.2 Datenübertragung

Der Sequence Chart in Abb. 3 zeigt den ungünstigsten Fall einer erfolgreichen Datenübertragung. Es wird angenommen, dass die für eine Datenübertragung notwendige Verbindung bereits aufgebaut ist (für die Testdatengenerierung wird TESDL entsprechend initialisiert).

In Abb. 3 ist eine erfolgreiche Datenübertragung mit dreimaligem Ablaufen des Timers gefolgt von einer halben Datenübertragung beschrieben. Die halbe Datenübertragung ist notwendig, weil das MDATind(AK) Signal zwischen dem Medium und dem Initiator nicht beobachtet werden kann. Der Empfang dieses Signals wird indirekt durch das zweite MDATind(DT)

bzw. IDATind nachgewiesen. Insgesamt treten an PCO ISAP1 zwei, an MSAP2 neun und an ISAP2 zwei Signale auf. Als Abbruchkriterien ergeben sich für die Testkonfiguration 1 eine Baumtiefe von vier und für die Testkonfigurationen 2 und 3 eine Baumtiefe von elf.

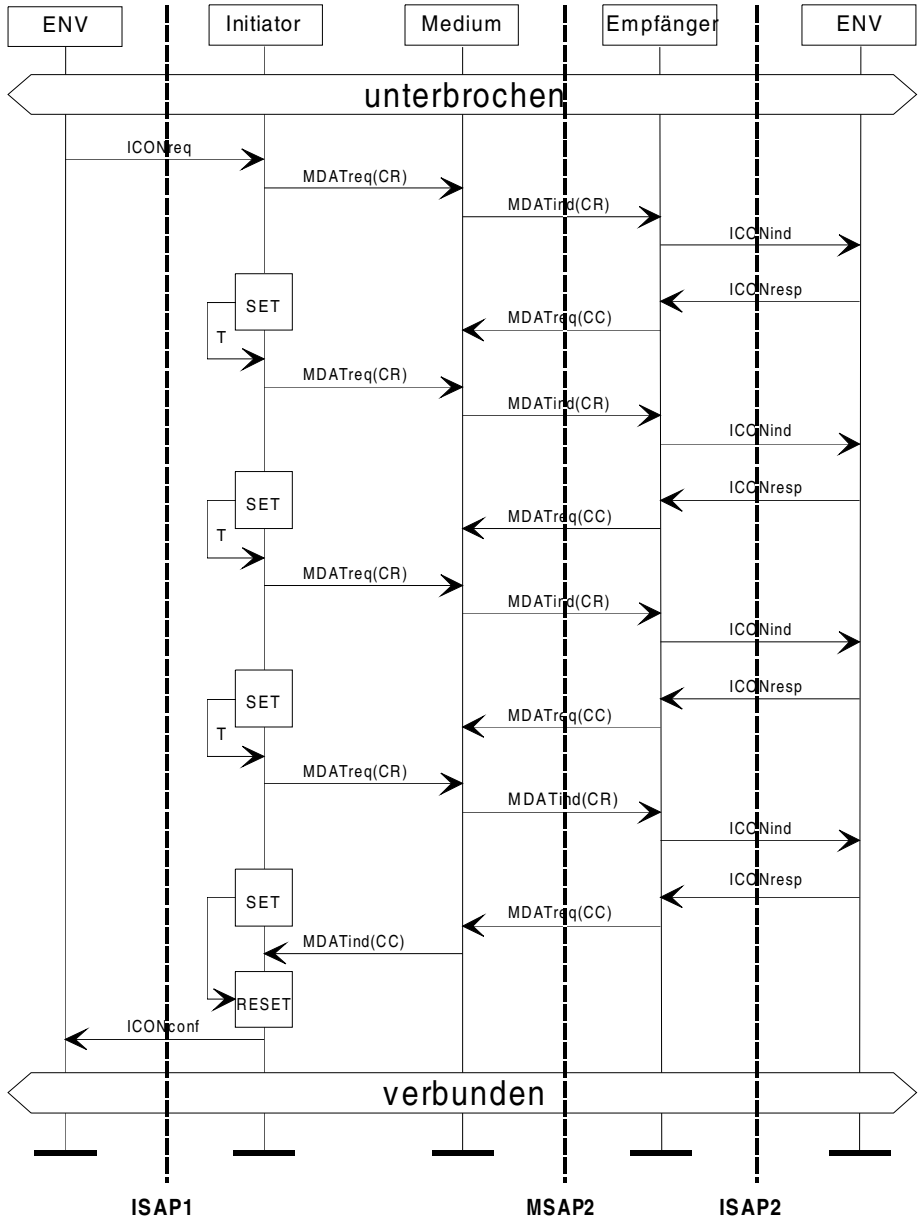


Abbildung 2: Der ungünstigste Fall eines erfolgreichen Verbindungsaufbaus

3.3 Verbindungsabbau

Der Sequence Chart in Abb. 4 zeigt einen Verbindungsabbau. Ein Verbindungsabbau kann in jedem Zustand des Inres Systems initiert werden. Bei den TESDL Experimenten wurde ange-

nommen, dass eine Verbindung aufgebaut ist. Für alle drei Testkonfigurationen ergibt sich eine Baumtiefe von zwei, da an jedem PCO nur ein Signal auftritt.

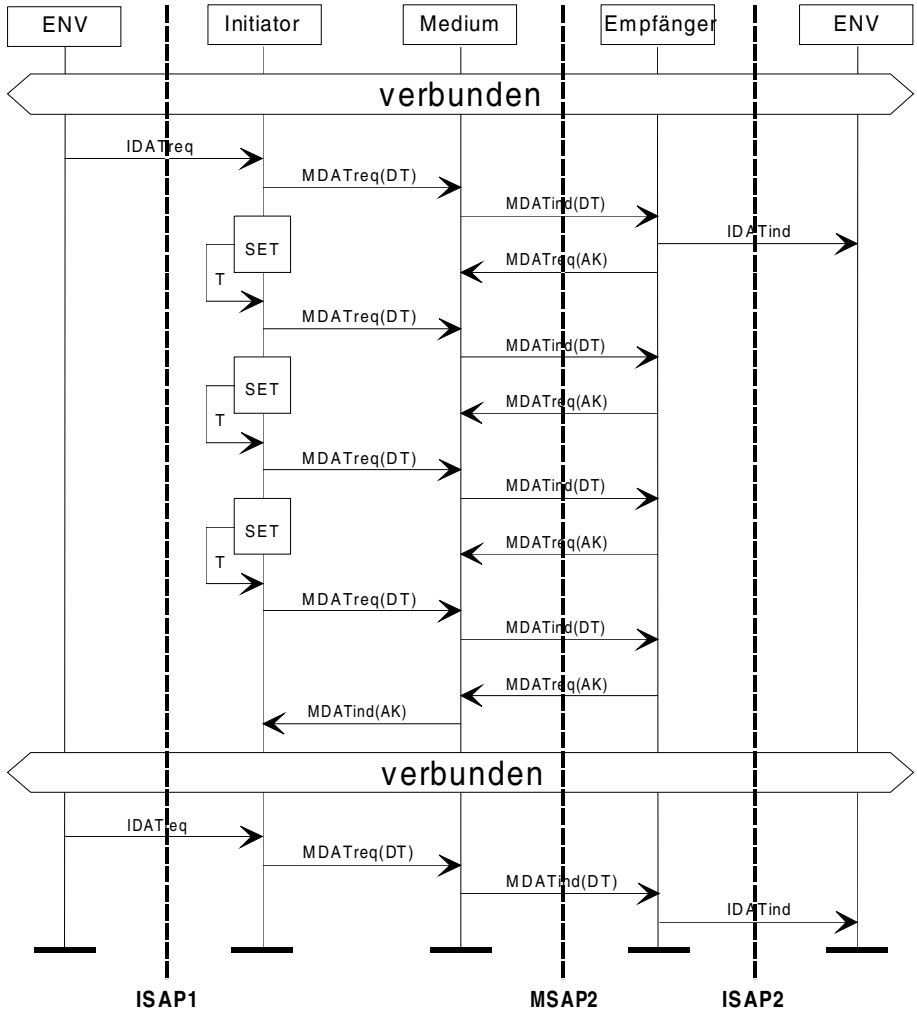


Abbildung 3: Der ungünstigste Fall einer erfolgreichen Datenübertragung

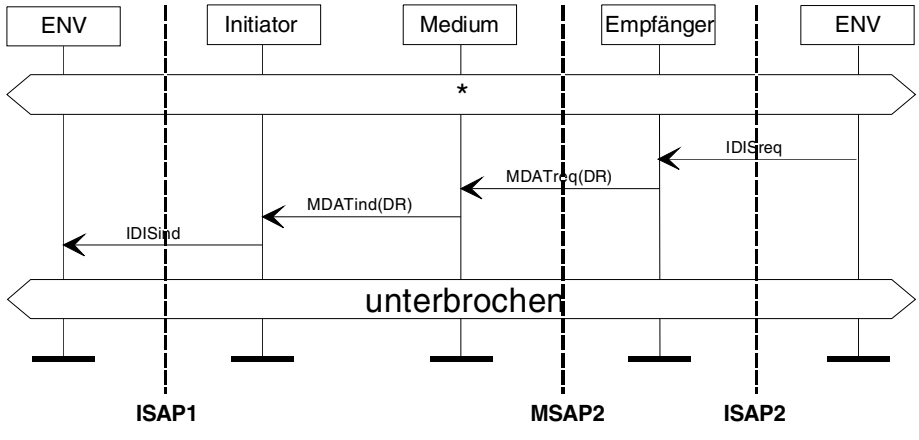


Abbildung 4: Verbindungsabbau

4 Ergebnisse der Testdatengenerierung mit TESDL

In diesem Abschnitt werden die Ergebnisse der Testdatengenerierung mit TESDL beschrieben und miteinander verglichen. Beispiele für die mit TESDL generierten TTCN Verhaltensbäume sind in Anhang B zu finden.

4.1 Testdatengenerierung für die drei Kommunikationsphasen

In den Abbildungen 5, 6 und 7 sind für die drei Kommunikationsphasen markante Zahlen für die generierten Testdaten einander gegenübergestellt. Hierbei muss beachtet werden, dass sich die Testkonfiguration 1 nicht direkt mit den Testkonfigurationen 2 und 3 vergleichen lässt, da die PCOs an unterschiedlichen Stellen liegen.

TESDL ermöglicht es, die automatische Generierung von bestimmten externen Signalen in vorher definierten Systemzuständen auszuschliessen. Um die Komplexität ein wenig einzuschränken, wurde bei den Experimenten für die Datenübertragung das Signal IDISreq in allen Zuständen ausgeschlossen. Dennoch konnte der TTCN Verhaltensbaum für die Datenübertragung bei Testkonfiguration 2 aufgrund der Komplexität (vgl. Abschnitt 4.2) nicht generiert werden. Das zugehörige TESDL Experiment wurde nach 6 Tagen ohne Ergebnis abgebrochen.

Verbindungsaufbau	Testkonfiguration 1	Testkonfiguration 2	Testkonfiguration 3
Baumtiefe	10	10	10
Generierungszeit	1d9h49'29"	2d8h19'56"	1h53'38"
Anzahl Zeilen	16835	131995	1639
Dateigrösse (Byte)	430182	4316060	49153
Anzahl Pfade	9965	88369	801

Abbildung 5: Resultate der TESDL Experimente für den Verbindungsaufbau

Datenübertragung	Testkonfiguration 1	Testkonfiguration 2	Testkonfiguration 3
Baumtiefe	4	11	11
Generierungszeit	2'37"	> 6d	46'41"
Anzahl Zeilen	34	*	375
Dateigrösse (Byte)	1596	*	14329
Anzahl Pfade	20	*	147

Abbildung 6: Resultate der TESDL Experimente für die Datenübertragung

Verbindungsabbau	Testkonfiguration 1	Testkonfiguration 2	Testkonfiguration 3
Baumtiefe	2	2	2
Generierungszeit	13"	23"	12"
Anzahl Zeilen	5	17	5
Dateigrösse (Byte)	645	977	670
Anzahl Pfade	3	13	3

Abbildung 7: Resultate der TESDL Experimente für den Verbindungsabbau

Baumtiefe	Generierungszeit	Dateigrösse (# Bytes)
2	19"	649
3	1'20"	1095
4	1'32"	1504
5	8'02"	4278
6	23'46"	9018
7	49'37"	21594
8	2h37'37"	48433
9	7h26'37"	122148

Abbildung 8: Resultate der qualitativen TESDL Experimente

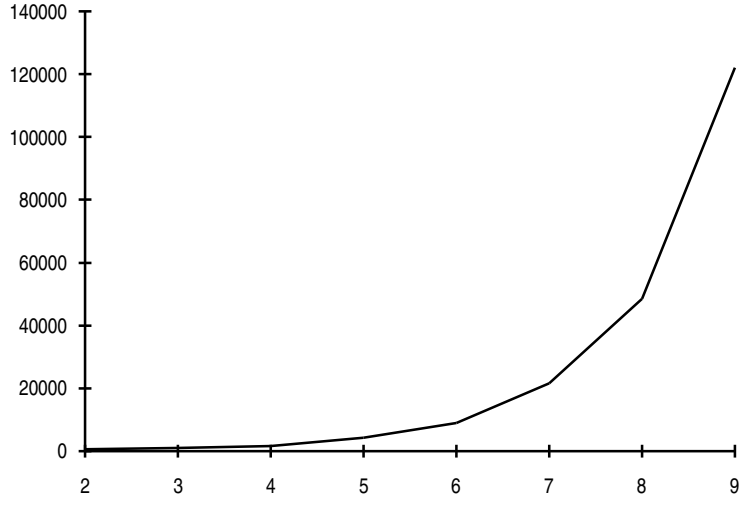


Abbildung 9: Dateigrösse (#Byte) in Abhängigkeit von der Baumtiefe

4.2 Komplexität der Testdatengenerierung

Bei diesen Messungen wird am Beispiel des Verbindungsaufbaus für Testkonfiguration 1 ein qualitativer Zusammenhang zwischen Generierungszeit und Baumtiefe sowie zwischen Dateigrösse der generierten Testdaten und Baumtiefe aufgezeigt. Wie man in den Abb. 8, 9 und 10 sieht, steigen Dateigrösse und Generierungszeit exponentiell gegenüber der Baumtiefe.

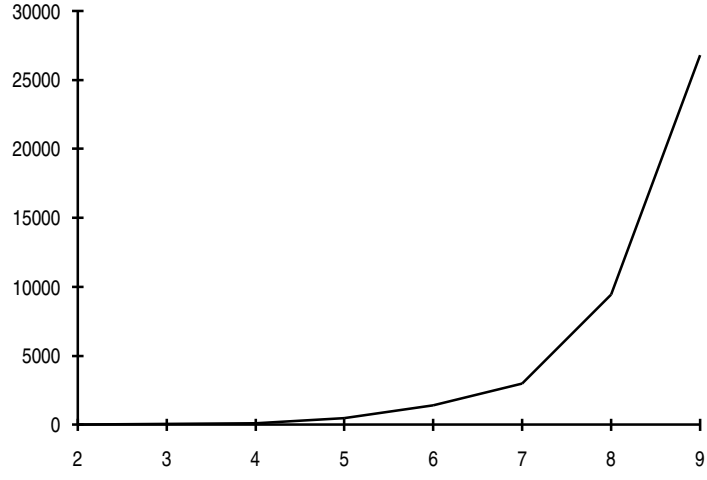


Abbildung 10: Generierungszeit (#Sekunden) in Abhängigkeit von der Baumtiefe

4.3 Bemerkungen zur Arbeit mit TESDL

Einige Bemerkungen zum Werkzeug und zur Arbeit mit TESDL:

- Bei der Modellierung des Inres Protokolls in SDL/PR musste auf die von TESDL unterstützten SDL Sprachkonstrukte Rücksicht genommen werden. So fehlt zum Beispiel das DECISION ANY Konstrukt und die Möglichkeit, Macros zu definieren.

- Es war trotz der verschiedenen Eingabemöglichkeiten, die TESDL bietet, nicht möglich, den TIMER des Initiators richtig zum laufen zu bringen. Deshalb wurde das Timersignal als externes Signal definiert, das von der Systemumgebung über einen eigenen Kanal und eine eigene Signalroute an den Initiator gesendet wird. TESDL erzeugt die Timersignale dann während der Simulation automatisch.

- Die zur Testdatengenerierung an TESDL übergebenen SDL/PR Eingabedaten werden nur einer sehr groben Syntaxprüfung unterzogen. So werden oft auch einfache Fehler erst beim Durchlaufen der entsprechenden Stelle bemerkt, also erst, wenn alle manuellen Eingaben gemacht sind und die Generierung der Testdaten bereits läuft.

- Das Benutzerinterface von TESDL ist nicht sehr bedienerfreundlich. Es sollte möglich sein, die Eingaben, die für jede Generierung manuell gemacht werden müssen, in einer Datei abzulegen.

- TESDL generiert nicht alle Sequenzen von Signalen, die nach Spezifikation möglich sind. So fehlt zum Beispiel bei den generierten Testdaten des Verbindungsaufbaus in der Testkonfiguration 3 der folgende Ablauf:

  ```
  (1)   ut!ICONreq
  (2)       lt!MDATreq(DR;...)
  ```

  Zudem ist ein grosser Teil der generierten Testdaten redundant, so zum Beispiel bei den generierten Daten der Datenübertragung in der Testkonfiguration 1. Dort sind nur die folgenden sechs Zeilen relevant:

  ```
  (1)   ut!IDATreq
  (2)       lt?IDATind
  (3)           ut?IDISind
  (4)               ut!ICONreq
  (3)           ut!IDATreq
  (4)               lt?IDATind
  ```

  TESDL generiert jedoch 34 Zeilen.

- Das Abbruchkriterium eines Testlaufes ist die Baumtiefe. Es ist nicht möglich, TESDL bei Erreichen eines bestimmten Zielzustandes die Testdatengenerierung abbrechen zu lassen.

- Die Dokumentation zu TESDL ist unvollständig. Einige der aufgeführten Beispiele liessen sich nicht nachvollziehen.

5 Die Entfernung von Redundanz aus TESDL Daten

Die mit TESDL generierten Testdaten sind, abhängig von der Komplexität des Kommunikationsprotokolls und der Baumtiefe, sehr umfangreich und zu einem großen Teil redundant. Dieses liegt daran, dass unterschiedliche (interne) Zustandsfolgen des SDL Systems das gleiche beobachtbare Verhalten haben. Hieraus ergeben sich identische Signalsequenzen. Da aber kein Einfluss darauf genommen werden kann, welche der möglichen Zustandsfolgen mit einer Signalsequenz durchlaufen wird, ist jedes doppelte Vorkommen der gleichen Sequenz für das Testen redundant. Es liegt daher nahe, redundante Signalsequenzen aus den mit TESDL generierten TTCN Verhaltensbäumen durch geeignete Reduktionsalgorithmen zu entfernen. Angaben zum Dialog mit der LISP Implementation der nachfolgend beschriebenen Algorithmen, sowie der zugehörige Programmcode und Beispiele für reduzierte TESDL Daten sind in den Anhängen C, D und E zu finden.

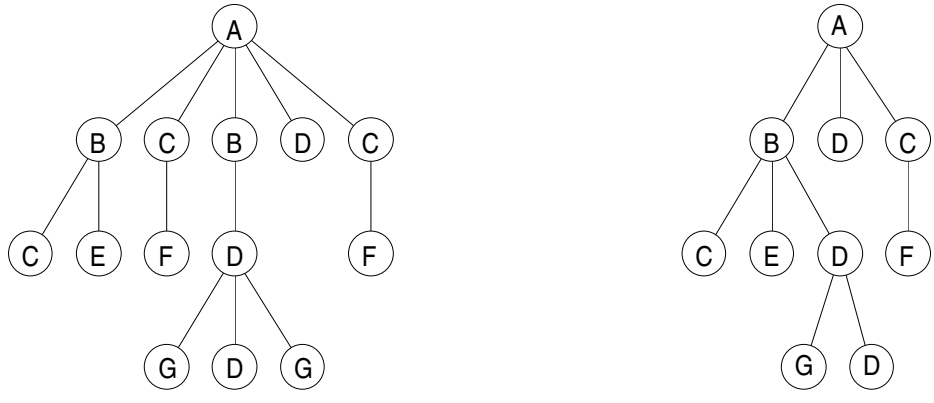
5.1 Algorithmen

Das Ziel der Algorithmen ist das Entfernen von redundanten Blattknoten und Unterbäumen aus den TTCN Verhaltensbäumen.

Grundalgorithmus (REDUC-1)

Das Entfernen der Redundanz aus den TTCN Verhaltensbäumen geschieht durch das Entfernen aller doppelten Blattknoten und dem Zusammenführen von Unterbäumen (Äste) mit gleichem Wurzelknoten. Alle identischen Signalsequenzen werden zu einer einzigen zusammengefasst (vgl. Abb. 11). Es werden folgende Schritte ausgeführt:

- Aufbau des TTCN Verhaltensbaumes während des Einlesens der Ereignisse (ein Ereignis repräsentiert hierbei das Senden oder Empfangen eines Signals) aus der Eingabedatei.
- Reduktion während des Baumaufbaus, d.h. jedes eingelesene Ereignis wird mit den auf der aktuellen Baumebene schon vorhandenen Ereignissen verglichen. Bei einer Übereinstimmung wird das eingelesene Ereignis ignoriert und über diesen Ast eine Ebene tiefer gestiegen. Ansonsten wird das Ereignis auf der aktuellen Ebene hinzugefügt.
- Ausgabe des reduzierten TTCN Verhaltensbaumes in die Ausgabedatei.



(a) Baum unreduziert (b) Baum aus (a) mit REDUC-1 reduziert

Abbildung 11: Baumreduktion mit REDUC-1

Modifizierter Algorithmus (REDUC-2)

Der Aufbau des Algorithmus von REDUC-2 ist weitgehend mit dem von REDUC-1 identisch, die Reduktionskriterien sind jedoch restriktiver. Nur doppelte und absolut identische Unterbäume werden zusammengeführt (vgl. Abb. 12).

Hierdurch bleiben Informationen bezüglich des beobachtbaren Verhaltens erhalten, die hinsichtlich des Testens eigentlich auch redundant sind. An Hand der resultierenden Verhaltensbäume lässt sich jedoch das beobachtbare Verhalten zweier Systeme unterscheiden.

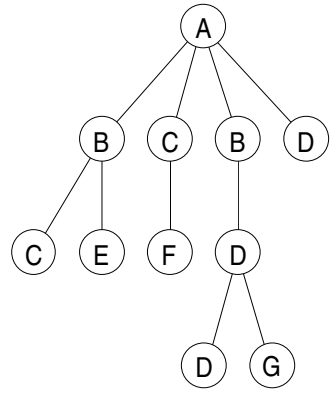


Abbildung 12: Baum von Abb. 1 (a) mit REDUC-2 reduziert

Implementation in LISP unter SUN OS

Für die Implementation der Algorithmen wurde aufgrund der Eignung zur Verarbeitung von Listen resp. Bäumen die Programmiersprache LISP ausgewählt. Wegen der Verfügbarkeit des LISP Interpreters und weil die Testdaten zum Teil sehr umfangreich sind, wurde die Implementation auf einer SUN Workstation durchgeführt.

Dateigrösse (#Zeilen)	Generierungszeit mit TESDL	Reduktionszeit REDUC-1	Reduktionszeit REDUC-2
6	19"	<1"	<1"
29	1'20"	<1"	<1"
49	1'32"	<1"	<1"
179	8'02"	<1"	1"
392	23'46"	1"	1"
928	49'37"	3"	3"
2036	2h37'37"	6"	6"
4933	7h26'37"	20"	14"
16835	1d9h49'29"	1'14"	1'12"
131995	2d8h19'56"	5'39"	6'07"

Abbildung 13: Zeitaufwand für Testdatengenerierung und Testdatenreduktion

5.2 Vergleich von Reduktions- und Generierungsaufwand

Die beiden Algorithmen wurden auf die von TESDL generierten Testdaten der verschiedenen Testkonfigurationen angewandt (vgl. Kapitel 4). Es zeigte sich, daß durch die Anwendung der Reduktionsalgorithmen der Umfang der Testdaten wesentlich verkleinert werden konnte. Der Zeitbedarf hielt sich dabei im Vergleich zur benötigten Zeit bei der Testdatengenerierung in einem bescheidenen Rahmen (Abb. 13).

Verbindungs-aufbau	Baumtiefe	Dateigrösse (# Zeilen)			Reduktion	
		TESDL	REDUC-1	REDUC-2	REDUC-1	REDUC-2
Testkonfiguration 1	10	16835	1324	1588	92%	91%
Testkonfiguration 2	10	131995	50185	113212	62%	14%
Testkonfiguration 3	10	1'639	432	894	74%	45%

Abbildung 14: Reduktion der TTCN Verhaltensbäume für den Verbindungsaufbau

Daten-übertragung	Baumtiefe	Dateigrösse (# Zeilen)			Reduktion	
		TESDL	REDUC-1	REDUC-2	REDUC-1	REDUC-2
Testkonfiguration 1	4	34	6	6	82%	82%
Testkonfiguration 2	Testdaten von TESDL nicht generiert (zu komplex)					
Testkonfiguration 3	11	375	121	135	68%	64%

Abbildung 15: Reduktion der TTCN Verhaltensbäume für die Datenübertragung

Verbindungs-abbau	Baumtiefe	Dateigrösse (# Zeilen)			Reduktion	
		TESDL	REDUC-1	REDUC-2	REDUC-1	REDUC-2
Testkonfiguration 1	2	5	5	5	0%	0%
Testkonfiguration 2	2	17	14	14	18%	18%
Testkonfiguration 3	2	5	4	4	20%	20%

Abbildung 16: Reduktion der TTCN Verhaltensbäume für den Verbindungsabbau

5.3 Reduktion der TESDL Daten für die Kommunikationsphasen

Die Tabellen in den Abb. 14, 15 und 16 vergleichen die reduzierten und die ursprünglichen Testdaten für die einzelnen Kommunikationsphasen (vgl. Abschnitt 4.1). Die Reduktion ist stark abhängig von der Dateigröße und beträgt bis zu 92 Prozent. Bedingt durch den restriktiveren Reduktionsalgorithmus ergibt sich für REDUC-2 im Vergleich zu REDUC-1 eine kleinere Reduktion (vgl. auch Abb. 17).

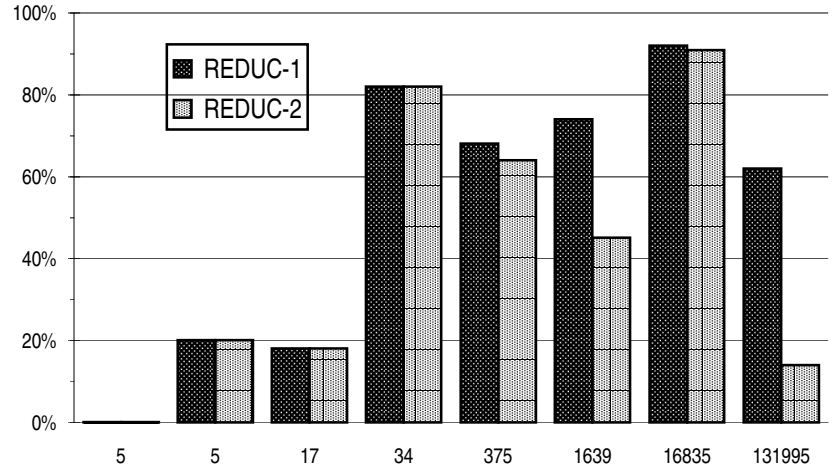


Abbildung 17: Reduktion in Abhängigkeit von der Dateigröße (#Zeilen)

Baumtiefe	Dateigrösse (# Zeilen)			Reduktion	
	TESDL	REDUC-1	REDUC-2	REDUC-1	REDUC-2
2	6	5	5	17%	17%
3	19	11	11	42%	42%
4	49	24	27	51%	45%
5	179	49	53	73%	70%
6	392	98	109	75%	72%
7	928	188	206	80%	78%
8	2036	360	398	82%	80%
9	4933	678	756	86%	85%
10	16835	1324	1588	92%	91%

Abbildung 18: Reduktion der TTCN Verhaltensbäume aus den qualitativen Messungen

5.4 Einflüsse der Reduktion auf die Speicherkomplexität

Die Tabellen in den Abb. 18 und 19 beschreiben die Ergebnisse der Reduktion der Testdaten aus den in Abschnitt 4.2 erläuterten Experimenten. Die Größe der reduzierten Testdaten in Abhängigkeit von der Baumtiefe der Testdaten steigt wesentlich langsamer.

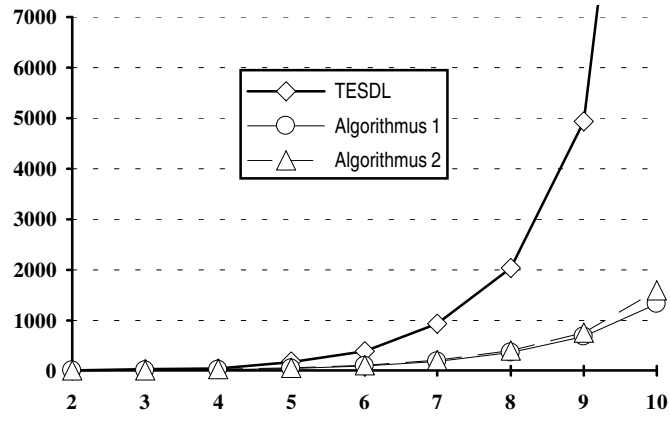


Abbildung 19: Grösse der Testdaten in Abhängigkeit von der Baumtiefe

6 Eine Interpretation der Experimente

Die Ergebnisse aller Experimente lassen sich bezüglich der Komplexität und bezüglich des Einflusses der verschiedenen Testkonfigurationen auf die Komplexität interpretieren.

Komplexität

Die Experimente mit TESDL haben gezeigt, dass die Generierung von TTCN Verhaltensbäumen aus SDL Spezifikationen eine in Abhängigkeit von der Baumtiefe exponentielle Speicher- und auch Zeitkomplexität besitzt. Abb. 12 zeigt, dass dieses für die Speicherkomplexität auch dann gilt, wenn man die für TESDL typische Redundanz aus den aus den generierten TTCN Verhaltensbäumen entfernt. An der Zeitkomplexität ändert sich durch die Reduktion nichts.

Der Einfluss der Testkonfigurationen auf die Komplexität

Ein Vergleich der Messergebnisse von Testkonfiguration 2 und 3 zeigt, dass der Bedarf an Speicherplatz und Zeit von Testkonfiguration 3 weniger komplex ist als der von Testkonfiguration 2. Die Testdatengenerierung für die Datenübertragung von Testkonfiguration 2 konnte sogar aufgrund des sehr grossen Zeitbedarfs nicht durchgeführt werden. Das entsprechende Experiment wurde nach 6 Tagen ohne Ergebnis abgebrochen. Der Unterschied zwischen den beiden Konfigurationen begründet sich dadurch, dass die Reihenfolge der an PCO MSAP2 beobachtbaren

Signale in Testkonfiguration 3 durch die Funktionalität des Empfängers eingeschränkt wird. Intuitiv lässt sich sagen, dass der bei Testkonfiguration 3 generierte TTCN Verhaltensbaum weniger breit ist, als der bei Testkonfiguration 2 generierte Baum.

Die Testkonfigurationen 1 und 3 unterscheiden sich durch die Beobachtung unterschiedliche PCOs. TESDL muss in beiden Konfigurationen Eingaben für die PCOs ISAP1 und ISAP2 erzeugen. Bei gleicher Baumtiefe begründet sich der höhere Platz- und Speicherbedarf von Testkonfiguration 1 darauf, dass bereits wenige Signale an den PCOs ISAP1 und ISAP2 eine grosse Menge von Signalen an PCO MSAP2 auslösen können. Das Abbruchkriterium bei der Testdatengenerierung für Testkonfiguration 3 wird schneller erreicht. Die Anzahl der Simulationspfade für Testkonfiguration 1 ist grösser, da im allgemeinen zur Erreichung des Abbruchkriteriums länger simuliert werden muss.

7 Zusammenfassung

In der vorliegenden Fallstudie ist an Hand des Inres Protokolls gezeigt worden, wie man mit Hilfe des Werkzeugs TESDL TTCN Verhaltensbäume für die unterschiedlichen Kommunikationsphasen eines verbindungsorientierten Protokolls generieren kann. Als Abbruchkriterium für die Baumgenerierung wurde die Anzahl der Signale ausgewählt, die im ungünstigsten Fall ausgetauscht werden muss, um die ausgewählte Kommunikationsphase erfolgreich zu durchlaufen. Ferner sind verschiedene Testkonfigurationen definiert und ihr Einfluss auf die Baumgenerierung untersucht worden. Daran anschliessend ist die für TESDL typische Redundanz aus den generierten TTCN Verhaltensbäumen entfernt und die dabei entstandenen redundanzfreien Bäume mit den Orginalen verglichen worden. Abschliessend wurden die Resultate aller Experimente noch einmal zusammenfassend interpretiert.

Danksagung

Die Autoren möchten Herrn Prof. Dieter Hogrefe für hilfreiche Diskussionen und die Unterstützung bei der Erstellung dieser Arbeit danken.

Literatur

[BrHoSa 91] Belina, F.; Hogrefe, D.; Sarma, A.: SDL with Applications from Protocol Specification, Prentice Hall International (UK) Ltd., 1991

[BrDoHo 90] Brömstrup, L.; Doan, Q.; Hogrefe, D.: Abschlussbericht zum Forschungsprojekt PROSPECT (Protokoll Specifikation und Test), im Auftrag der Deutschen Bundespost, Universität Hamburg, Fachbereich Informatik, 1990

[Brö 89] Brömstrup, L.: Conformance Testen: Automatische Testfallgenerierung aus SDL-Spezifikationen, Diplomarbeit an der Universität Hamburg, 1989

[BröHog 89] Brömstrup, L.; Hogrefe, D.: TESDL: Experience with Generating Test Cases from SDL Specifications, in SDL'89 The Language at Work - O. Faergemand and M. M. Marques (Editors), North-Holland, 1989

[GraRu 89] Grabowski, J.; Rudolph, E.: Putting Extended Sequence Charts to Practice, SDL'89 The Language at Work - O. Faergemand and M. M. Marques (editors), North-Holland, 1989

[Hog 88] Hogrefe, D.: Automatic Generation of Test Cases from SDL Specifications, CCITT SDL Newsletter 12, 1988

[Hog 89] Hogrefe, D.: Estelle, LOTOS und SDL: Standard-Spezifikationssprachen für verteilte Systeme, Springer Verlag, 1989

[Hog 92] Hogrefe, D.: OSI Formal Specification Case Study: the Inres Protocol and Service (revised), Technischer Bericht IAM-91-012, Universität Bern, 1992

[ISO 7498] ISO/IEC TC97: Basic Reference Model, International Standard 7498, 1984

[ISO 9646-3] ISO/IEC: Information Technology - Opens Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation, International Standard 9646-3, 1991

[ISO 10167] ISO/IEC TC97/SC21: Guidelines for the Application of Estelle, LOTOS and SDL, TR 10167, 1990

[PROS 90] Anhang zum Abschlussbericht Forschungsprojekt PROSPECT, 1990

[Z100] CCITT Recommendation Z.100: Specification and Description Language (SDL), Genf, 1992

[Z120] CCITT Recommendation Z.120: Message Sequence Chart (MSC), Genf, 1992

Anhänge

Anhang A SDL Spezifikationen

A.1 SDL Spezifikationen für die Testkonfigurationen 1 und 3

```
system INRES_Protokoll;

  newtype ISDUTyp
    struct
      Data1 integer;
      Data2 integer;
  endnewtype ISDUTyp;

  newtype MSDUTyp
    struct
      id    charstring;
      Num   boolean;
      Daten ISDUTyp;
  endnewtype MSDUTyp;

  signal
      ICONreq,
      IDATreq( ISDUTyp),
      ICONconf,
      ICONind,
      ICONresp,
      IDISreq,
      IDISind,
      IDATind( ISDUTyp),
      MDATreq( MSDUTyp),
      MDATind( MSDUTyp),
      T,
      AK1;

  channel ISAPiniT
    from env to Station_Ini
      with T;
  endchannel ISAPiniT;

  channel ISAPEmpfAK
    from env to Station_Empf
      with AK1;
  endchannel ISAPEmpfAK;

  channel ISAPini
    from env to Station_Ini
      with ICONreq, IDATreq;
    from Station_Ini to env
      with ICONconf, IDISind;
  endchannel ISAPini;

  channel ISAPempf
    from env to Station_Empf
      with ICONresp, IDISreq;
    from Station_Empf to env
      with ICONind, IDATind;
  endchannel ISAPempf;

  channel MSAP1
    from Station_Ini to Medium
      with MDATreq;
    from Medium to Station_Ini
      with MDATind;
  endchannel MSAP1;

  channel MSAP2
    from Station_Empf to Medium
      with MDATreq;
    from Medium to Station_Empf
      with MDATind;
  endchannel MSAP2;

  block Station_Ini  referenced;
  block Station_Empf referenced;
  block Medium       referenced;

endsystem INRES_Protokoll;


block Station_Ini;

  signal
      CC,
      AK( boolean),
      DR, CR,
      DT( boolean, ISDUTyp);

  connect ISAPiniT and ISAP_T;
  connect ISAPini and ISAP;
  connect MSAP1 and MSAP;

  signalroute ISAP_T
    from env to Initiator
      with T;

  signalroute ISAP
    from env to Initiator
      with ICONreq, IDATreq;
    from Initiator to env
      with ICONconf, IDISind;

  signalroute MSAP
    from env to Codierer_ini
      with MDATind;
    from Codierer_ini to env
      with MDATreq;

  signalroute IPDU
    from Initiator to Codierer_ini
      with CR, DT;
    from Codierer_ini to Initiator
      with CC, AK, DR;

  process Initiator  ( 1, 1) referenced;
  process Codierer_Ini ( 1, 1) referenced;

endblock Station_Ini;


block Station_Empf;

  signal
      CC,
      AK( boolean),
      DR, CR,
      DT( boolean, ISDUTyp);

  connect ISAPempf and ISAP;
  connect MSAP2 and MSAP;
  connect ISAPEmpfAK and ISAPAK;

  signalroute ISAPAK
    from env to Empfaenger
      with AK1;

  signalroute ISAP
    from env to Empfaenger
      with ICONresp, IDISreq;
    from Empfaenger to env
      with ICONind, IDATind;

  signalroute MSAP
    from env to Codierer_Empf
      with MDATind;
    from Codierer_Empf to env
      with MDATreq;

  signalroute IPDU
    from Empfaenger to Codierer_Empf
      with CC, AK, DR;
    from Codierer_Empf to Empfaenger
      with CR, DT;

  process Empfaenger    ( 1, 1) referenced;
  process Codierer_Empf ( 1, 1) referenced;

endblock Station_Empf;


block Medium;

  signal
      IDAT( MSDUTyp);

  connect MSAP1 and MSAP1M;
  connect MSAP2 and MSAP2M;

  signalroute MSAP1M
    from env to MSAP_Mgr_Ini
      with MDATreq;
    from MSAP_Mgr_Ini to env
      with MDATind;

  signalroute MSAP2M
    from env to MSAP_Mgr_Empf
      with MDATreq;
    from MSAP_Mgr_Empf to env
      with MDATind;

  signalroute Intern
    from MSAP_Mgr_Ini to MSAP_Mgr_Empf
      with IDAT;
    from MSAP_Mgr_Empf to MSAP_Mgr_Ini
      with IDAT;

  process MSAP_Mgr_Ini  ( 1, 1) referenced;
  process MSAP_Mgr_Empf ( 1, 1) referenced;

endblock Medium;


process Initiator ( 1, 1);

 dcl
   Zaehler integer,
   d       ISDUTyp,
   Num     boolean,
   Nummer  boolean;

 start;
   task Zaehler := 1;
   task Nummer := TRUE;
   task Num := TRUE;
   nextstate unterbrochen;

 state unterbrochen;
   input DR;
     output IDISind;
     nextstate unterbrochen;
   input ICONreq;
     task Zaehler := 1;
     output CR;
     nextstate warten;

   state warten;
```

```
    input CC;
      task Nummer := TRUE;
      output ICONconf;
      nextstate verbunden;
    input T;
      decision Zaehler;
      ( < 4) :
        output CR;
        task Zaehler := Zaehler + 1;
        nextstate warten;
      else :
        output IDISind;
        nextstate unterbrochen;
      enddecision;
    input DR;
      output IDISind;
      nextstate unterbrochen;

  state verbunden;
    input IDATreq( d);
      output DT( Nummer, d);
      task Zaehler := 1;
      nextstate sendend;
    input DR;
      output IDISind;
      nextstate unterbrochen;

  state sendend;
    input AK( Num);
      decision Num;
      ( = Nummer ) :
        task Nummer := Nummer XOR TRUE;
        nextstate verbunden;
      else :
        label1 :
        decision Zaehler;
        ( < 4) :
          output DT( Nummer, d);
          task Zaehler := Zaehler + 1;
          nextstate sendend;
        else :
          output IDISind;
          nextstate unterbrochen;
        enddecision;
      enddecision;
    input T;
      join label1;

endprocess Initiator;


process Empfaenger ( 1, 1);

  dcl
    d     ISDUTyp,
    hilf  boolean,
    Num   boolean,
    Nummer boolean;

  start;
    task Nummer := FALSE;
    task Num := TRUE;
    nextstate unterbrochen;

  state unterbrochen;
    input CR;
      output ICONind;
      nextstate warten;

  state warten;
    input ICONresp;
      task Nummer := FALSE;
      output CC;
      nextstate verbunden;

  state verbunden;
    input CR;
      output ICONind;
      nextstate warten;
    input DT( Num, d);
      task hilf := Nummer XOR TRUE;
      decision Num;
      ( = hilf ):
        output IDATind( d);
        task Nummer := Nummer XOR TRUE;
        nextstate verbunden1;
      else :
        nextstate verbunden1;

  state verbunden1;
    input CR;
      output ICONind;
      nextstate warten;
    input AK1;
      output AK( Num);
      nextstate verbunden;
    input DT( Num, d);
      task hilf := Nummer XOR TRUE;
      decision Num;
      ( = hilf ):
        output IDATind( d);
        task Nummer := Nummer XOR TRUE;
        nextstate verbunden1;
      else :
        nextstate verbunden1;

  state *;
    input IDISreq;
      output DR;
      nextstate unterbrochen;

endprocess Empfaenger;


process Codierer_Ini ( 1, 1);

  dcl
    d   ISDUTyp,
    Num boolean,
    Sdu MSDUTyp;

  start;
    nextstate bereit;

  state bereit;
    input CR;
      task Sdu!id := 'CR';
      label2 :
      output MDATreq( Sdu);
      nextstate bereit;
    input DT( Num, d);
      task Sdu!id := 'DT';
      task Sdu!Num := Num;
      task Sdu!Daten := d;
      join label2;
    input MDATind( Sdu);
      decision Sdu!id;
      ( = 'CC') :
        output CC;
        nextstate bereit;
      ( = 'AK') :
        output AK( Sdu!Num);
        nextstate bereit;
      ( = 'DR') :
        output DR;
        nextstate bereit;
      else :
        nextstate bereit;
      enddecision;

endprocess Codierer_Ini;


process Codierer_Empf ( 1, 1);

  dcl
    Num boolean,
    Sdu MSDUTyp;

  start;
    nextstate bereit;

  state bereit;
    input DR;
      task Sdu!id := 'DR';
      label3 :
      output MDATreq( Sdu);
      nextstate bereit;
    input CC;
      task Sdu!id := 'CC';
      join label3;
    input AK( Num);
      task Sdu!id := 'AK';
      task Sdu!Num := Num;
      join label3;
    input MDATind( Sdu);
      decision Sdu!id;
      ( = 'CR') :
        output CR;
        nextstate bereit;
      ( = 'DT') :
        output DT( Sdu!Num, Sdu!Daten);
        nextstate bereit;
      else :
        nextstate bereit;
      enddecision;

endprocess Codierer_Empf;


process MSAP_Mgr_Ini ( 1, 1);

  dcl
    d MSDUTyp;

  start;
    nextstate bereit;

  state bereit;
    input MDATreq( d);
      output IDAT( d);
      nextstate bereit ;
    input IDAT( d);
      output MDATind( d);
      nextstate bereit;

endprocess MSAP_Mgr_Ini;


process MSAP_Mgr_Empf ( 1, 1);

  dcl
    d MSDUTyp;

  start;
    nextstate bereit;

  state bereit;
    input MDATreq( d);
      output IDAT( d);
      nextstate bereit;
    input IDAT( d);
      output MDATind( d);
      nextstate bereit;

endprocess MSAP_Mgr_Empf;
```

A.2 SDL Spezifikation für Testkonfiguration 2

```
system INRES_Protokoll;

  newtype ISDUTyp
    struct
      Data1 integer;
      Data2 integer;
  endnewtype ISDUTyp;

  newtype MSDUTyp
    struct
      id    charstring;
      Num   boolean;
      Daten ISDUTyp;
  endnewtype MSDUTyp;

  signal
      ICONreq,
      IDATreq( ISDUTyp),
      ICONconf,
      IDISind,
      MDATreq( MSDUTyp),
      MDATind( MSDUTyp),
      CC,
      AK( boolean),
      DR,
      CR,
      DT( boolean, ISDUTyp),
      T;

  channel ISAPiniT
    from env to Station_Ini
      with T;
  endchannel ISAPiniT;

  channel ISAPini
    from env to Station_Ini
      with ICONreq, IDATreq;
    from Station_Ini to env
      with ICONconf, IDISind;
  endchannel ISAPini;

  channel ISAPempf
    from env to Station_Empf
      with CC, AK, DR;
    from Station_Empf to env
      with CR, DT;
  endchannel ISAPempf;

  channel MSAP1
    from Station_Ini to Medium
      with MDATreq;
    from Medium to Station_Ini
      with MDATind;
  endchannel MSAP1;

  channel MSAP2
    from Station_Empf to Medium
      with MDATreq;
    from Medium to Station_Empf
      with MDATind;
  endchannel MSAP2;

  block Station_Ini  referenced;
  block Station_Empf referenced;
  block Medium       referenced;

endsystem INRES_Protokoll;


block Station_Ini;

  signal
      CC,
      AK( boolean),
      DR, CR,
      DT( boolean, ISDUTyp);

  connect ISAPiniT and ISAP_T;
  connect ISAPini and ISAP;
  connect MSAP1 and MSAP;

  signalroute ISAP_T
    from env to Initiator
      with T;

  signalroute ISAP
    from env to Initiator
      with ICONreq, IDATreq;
    from Initiator to env
      with ICONconf, IDISind;

  signalroute MSAP
    from env to Codierer_ini
      with MDATind;
    from Codierer_ini to env
      with MDATreq;

  signalroute IPDU
    from Initiator to Codierer_ini
      with CR, DT;
    from Codierer_ini to Initiator
      with CC, AK, DR;

  process Initiator  ( 1, 1) referenced;
  process Codierer_Ini ( 1, 1) referenced;

endblock Station_Ini;


block Station_Empf;

  connect ISAPempf and ISAP;
  connect MSAP2 and MSAP;

  signalroute ISAP
    from env to Codierer_Empf
      with CC, AK, DR;
    from Codierer_Empf to env
      with CR, DT;

  signalroute MSAP
    from env to Codierer_Empf
      with MDATind;
    from Codierer_Empf to env
      with MDATreq;

  process Codierer_Empf ( 1, 1) referenced;

endblock Station_Empf;


block Medium;

  signal
      IDAT( MSDUTyp);

  connect MSAP1 and MSAP1M;
  connect MSAP2 and MSAP2M;

  signalroute MSAP1M
    from env to MSAP_Mgr_Ini
      with MDATreq;
    from MSAP_Mgr_Ini to env
      with MDATind;

  signalroute MSAP2M
    from env to MSAP_Mgr_Empf
      with MDATreq;
    from MSAP_Mgr_Empf to env
      with MDATind;

  signalroute Intern
    from MSAP_Mgr_Ini to MSAP_Mgr_Empf
      with IDAT;
    from MSAP_Mgr_EMPF to MSAP_Mgr_Ini
      with IDAT;

  process MSAP_Mgr_Ini  ( 1, 1) referenced;
  process MSAP_Mgr_Empf ( 1, 1) referenced;

endblock Medium;


process Initiator ( 1, 1);

  dcl
    Zaehler integer,
    d       ISDUTyp,
    Num     boolean,
    Nummer  boolean;

  start;
    task Zaehler := 1;
    task Nummer := TRUE;
    task Num := TRUE;
    nextstate unterbrochen;

  state unterbrochen;
    input DR;
      output IDISind;
      nextstate unterbrochen;
    input ICONreq;
      task Zaehler := 1;
      output CR;
      nextstate warten;

    state warten;
      input CC;
        task Nummer := TRUE;
        output ICONconf;
        nextstate verbunden;
      input T;
        decision Zaehler;
        ( < 4 ) :
          output CR;
          task Zaehler := Zaehler + 1;
          nextstate warten;
        else :
          output IDISind;
          nextstate unterbrochen;
        enddecision;
      input DR;
        output IDISind;
        nextstate unterbrochen;

    state verbunden;
      input IDATreq( d);
        output DT( Nummer, d);
        task Zaehler := 1;
        nextstate sendend;
      input DR;
        output IDISind;
```

```
      nextstate unterbrochen;

  state sendend;
    input AK( Num);
      decision Num;
      ( = Nummer ) :
        task Nummer := Nummer XOR TRUE;
        nextstate verbunden;
      else :
        label1 :
        decision Zaehler;
        ( < 4 ) :
          output DT( Nummer, d);
          task Zaehler := Zaehler + 1;
          nextstate sendend;
        else :
          output IDISind;
          nextstate unterbrochen;
        enddecision;
      enddecision;
    input T;
      join label1;

endprocess Initiator;


process Codierer_Ini ( 1, 1);

  dcl
    d   ISDUTyp,
    Num boolean,
    Sdu MSDUTyp;

  start;
    nextstate bereit;

  state bereit;
    input CR;
      task Sdu!id := 'CR';
      label2 :
      output MDATreq( Sdu);
      nextstate bereit;
    input DT( Num, d);
      task Sdu!id := 'DT';
      task Sdu!Num := Num;
      task Sdu!Daten := d;
      join label2;
    input MDATind( Sdu);
      decision Sdu!id;
      ( = 'CC') :
        output CC;
        nextstate bereit;
      ( = 'AK') :
        output AK( Sdu!Num);
        nextstate bereit;
      ( = 'DR') :
        output DR;
        nextstate bereit;
      else :
        nextstate bereit;
      enddecision;

endprocess Codierer_Ini;


process Codierer_Empf ( 1, 1);

  dcl
    Num boolean,
    Sdu MSDUTyp;

  start;
    nextstate bereit;

  state bereit;
    input DR;
      task Sdu!id := 'DR';
      label3 :
      output MDATreq( Sdu);
      nextstate bereit;
    input CC;
      task Sdu!id := 'CC';
      join label3;
    input AK( Num);
      task Sdu!id := 'AK';
      task Sdu!Num := Num;
      join label3;
    input MDATind( Sdu);
      decision Sdu!id;
      ( = 'CR') :
        output CR;
        nextstate bereit;
      ( = 'DT') :
        output DT( Sdu!Num, Sdu!Daten);
        nextstate bereit;
      else :
        nextstate bereit;
      enddecision;

endprocess Codierer_Empf;


process MSAP_Mgr_Ini ( 1, 1);

  dcl
    d MSDUTyp;

  start;
    nextstate bereit;

  state bereit;
    input MDATreq( d);
      output IDAT( d);
      nextstate bereit ;
    input IDAT( d);
      output MDATind( d);
      nextstate bereit;

endprocess MSAP_Mgr_Ini;


process MSAP_Mgr_Empf ( 1, 1);

  dcl
    d MSDUTyp;

  start;
    nextstate bereit;

  state bereit;
    input MDATreq( d);
      output IDAT( d);
      nextstate bereit;
    input IDAT( d);
      output MDATind( d);
      nextstate bereit;

endprocess MSAP_Mgr_Empf;
```

Anhang B Mit TESDL generierte TTCN Verhaltensbäume

Wegen des Umfangs der generierten Testdaten, vgl. Tabellen im Kapitel 4.1, können hier nicht alle Daten ausgedruckt werden. Folgende wurden weggelassen: Der Verbindungsaufbau für die Testkonfigurationen 1 und 2 und die Datenphase für die Testkonfiguration 2.

B.1 Verbindungsaufbau für die Testkonfiguration 3

TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 10
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-12 19:42:28

```
( 1) ut!ICONreq
( 2)  lt?MDATind[CR,,,]
( 3)  lt?MDATind[CR,,,]
( 4)   lt?MDATind[CR,,,]
( 5)   lt?MDATind[CR,,,]
( 6)    ut?IDISind
( 7)    ut!ICONreq
( 8)     lt?MDATind[CR,,,]
( 9)     lt?MDATind[CR,,,]
(10)      lt?MDATind[CR,,,]
(10)      lt!MDATreq[CC,,,]
(10)      lt!MDATreq[DR,,,]
( 9)     lt!MDATreq[CC,,,]
(10)      ut?ICONconf
(10)      ut?ICONconf
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 7)    lt!MDATreq[CC,,,]
( 8)     ut!ICONreq
( 9)     lt?MDATind[CR,,,]
(10)      lt?MDATind[CR,,,]
(10)      lt!MDATreq[CC,,,]
(10)      lt!MDATreq[DR,,,]
(10)      lt?MDATind[CR,,,]
( 8)     lt!MDATreq[DR,,,]
( 9)      ut?IDISind
(10)      ut!ICONreq
(10)      lt!MDATreq[DR,,,]
( 9)      ut?IDISind
(10)      lt!MDATreq[DR,,,]
( 7)    lt!MDATreq[DR,,,]
( 8)     ut?IDISind
( 9)     ut!ICONreq
(10)      lt?MDATind[CR,,,]
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 8)     ut?IDISind
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 6)    ut?IDISind
( 7)    lt!MDATreq[CC,,,]
( 8)     ut!ICONreq
( 9)     lt?MDATind[CR,,,]
(10)      lt?MDATind[CR,,,]
(10)      lt!MDATreq[CC,,,]
(10)      lt!MDATreq[DR,,,]
(10)      lt!MDATreq[DR,,,]
(10)      lt?MDATind[CR,,,]
( 8)     lt!MDATreq[DR,,,]
( 9)      ut?IDISind
(10)      ut!ICONreq
(10)      lt!MDATreq[DR,,,]
( 9)      ut?IDISind
(10)      lt!MDATreq[DR,,,]
( 6)    ut?IDISind
( 7)    lt!MDATreq[DR,,,]
( 8)     ut?IDISind
( 9)     ut!ICONreq
(10)      lt?MDATind[CR,,,]
(10)      lt!MDATreq[DR,,,]
( 9)     lt?MDATind[CR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 8)     ut?IDISind
( 9)      lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 7)    lt!MDATreq[DR,,,]
( 8)     ut?IDISind
( 9)     ut!ICONreq
(10)      lt?MDATind[CR,,,]
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 6)    ut?IDISind
( 7)    lt!MDATreq[DR,,,]
( 8)     ut?IDISind
( 9)     ut!ICONreq
(10)      lt?MDATind[CR,,,]
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 8)     ut?IDISind
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 6)    ut?IDISind
( 7)    lt!MDATreq[DR,,,]
( 8)     ut?IDISind
( 9)     ut!ICONreq
(10)      lt?MDATind[CR,,,]
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 8)     ut?IDISind
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
```

```
( 4)   lt!MDATreq[CC,,,]
( 5)    ut?ICONconf
( 6)     ut!IDATreq[,0,0]
( 7)      lt?MDATind[DT,TRUE,0,0]
( 8)       lt?MDATind[DT,TRUE,0,0]
( 9)        lt?MDATind[DT,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[AK,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 9)        lt!MDATreq[AK,TRUE,0,0]
(10)         ut!IDATreq[,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 9)        lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 8)       lt!MDATreq[AK,TRUE,0,0]
( 9)        ut!IDATreq[,0,0]
(10)         lt?MDATind[DT,FALSE,0,0]
( 9)        lt!MDATreq[DR,TRUE,0,0]
(10)         ut?IDISind
(10)         ut?IDISind
( 8)       lt!MDATreq[DR,TRUE,0,0]
( 9)        lt?MDATind[DT,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 9)        lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 8)       lt?MDATind[DT,TRUE,0,0]
( 9)        lt?MDATind[DT,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[AK,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 9)        lt!MDATreq[AK,TRUE,0,0]
(10)         ut!IDATreq[,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 9)        lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 6)     lt!MDATreq[DR,,,]
( 7)      ut?IDISind
( 8)       ut!ICONreq
( 9)        lt?MDATind[CR,,,]
(10)         lt?MDATind[CR,,,]
(10)         lt!MDATreq[CC,,,]
(10)         lt!MDATreq[DR,,,]
(10)         lt?MDATind[CR,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)         ut!ICONreq
(10)         lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)         lt!MDATreq[DR,,,]
( 7)      ut?IDISind
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)         ut!ICONreq
(10)         lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)         lt!MDATreq[DR,,,]
( 5)     ut?ICONconf
( 6)      lt!MDATreq[DR,,,]
( 7)       ut?IDISind
( 8)        ut!ICONreq
( 9)         lt?MDATind[CR,,,]
(10)          lt?MDATind[CR,,,]
(10)          lt!MDATreq[CC,,,]
(10)          lt!MDATreq[DR,,,]
(10)          lt?MDATind[CR,,,]
( 8)        lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 7)       ut?IDISind
( 8)        lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 4)    lt!MDATreq[DR,,,]
( 5)     ut?IDISind
( 6)      ut!ICONreq
( 7)       lt?MDATind[CR,,,]
( 8)        lt?MDATind[CR,,,]
( 9)         lt?MDATind[CR,,,]
(10)          lt?MDATind[CR,,,]
(10)          lt!MDATreq[CC,,,]
(10)          lt!MDATreq[DR,,,]
(10)          lt?MDATind[CR,,,]
( 9)         lt!MDATreq[CC,,,]
(10)          lt?MDATind[CR,,,]
(10)          lt!MDATreq[CC,,,]
(10)          lt!MDATreq[DR,,,]
( 9)         lt!MDATreq[DR,,,]
(10)          lt?MDATind[CR,,,]
( 8)        lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 7)       ut?IDISind
( 8)        lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 6)      lt!MDATreq[DR,,,]
( 7)       ut?IDISind
( 8)        ut!ICONreq
( 9)         lt?MDATind[CR,,,]
(10)          lt?MDATind[CR,,,]
(10)          lt!MDATreq[CC,,,]
(10)          lt!MDATreq[DR,,,]
(10)          lt?MDATind[CR,,,]
( 8)        lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 7)       ut?IDISind
( 8)        lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 3)    lt!MDATreq[CC,,,]
( 4)     ut?ICONconf
( 5)      ut!IDATreq[,0,0]
( 6)       lt?MDATind[DT,TRUE,0,0]
( 7)        lt?MDATind[DT,TRUE,0,0]
( 8)         lt?MDATind[DT,TRUE,0,0]
( 9)          lt?MDATind[DT,TRUE,0,0]
(10)           ut?IDISind
(10)           ut?IDISind
(10)           ut?IDISind
(10)           lt!MDATreq[AK,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 9)          lt!MDATreq[AK,TRUE,0,0]
(10)           ut!IDATreq[,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 9)          lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 8)         lt!MDATreq[AK,TRUE,0,0]
( 9)          ut!IDATreq[,0,0]
(10)           lt?MDATind[DT,FALSE,0,0]
( 9)          lt!MDATreq[DR,TRUE,0,0]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         lt!MDATreq[DR,TRUE,0,0]
( 9)          lt?MDATind[DT,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 9)          lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 8)         lt!MDATreq[AK,TRUE,0,0]
( 9)          ut!IDATreq[,0,0]
(10)           lt?MDATind[DT,FALSE,0,0]
( 9)          lt!MDATreq[DR,TRUE,0,0]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         lt!MDATreq[DR,TRUE,0,0]
( 9)          lt?MDATind[DT,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 9)          lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt!MDATreq[DR,,,]
( 9)          lt!MDATreq[CC,,,]
(10)           ut?ICONconf
(10)           ut?ICONconf
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         lt!MDATreq[DR,,,]
( 9)          lt?MDATind[CR,,,]
(10)           lt?MDATind[CR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt!MDATreq[DR,,,]
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 3)    lt!MDATreq[DR,,,]
( 4)     ut?IDISind
( 5)      ut!ICONreq
( 6)       lt?MDATind[CR,,,]
( 7)        lt?MDATind[CR,,,]
( 8)         lt?MDATind[CR,,,]
( 9)          lt?MDATind[CR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
(10)           ut?IDISind
(10)           lt!MDATreq[CC,,,]
(10)           lt!MDATreq[DR,,,]
( 9)          lt!MDATreq[CC,,,]
(10)           ut?ICONconf
(10)           ut?ICONconf
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         lt!MDATreq[CC,,,]
( 9)          ut?ICONconf
```

```
(10)       ut!IDATreq[,0,0]
(10)       lt!MDATreq[DR,,,]
( 9)      ut?ICONconf
(10)       lt!MDATreq[DR,,,]
( 8)     lt!MDATreq[DR,,,]
( 9)      ut?IDISind
(10)       ut!ICONreq
(10)       lt!MDATreq[DR,,,]
( 9)      ut?IDISind
(10)       lt!MDATreq[DR,,,]
( 7)    lt!MDATreq[CC,,,]
( 8)     ut?ICONconf
( 9)      ut!IDATreq[,0,0]
(10)       lt?MDATind[DT,TRUE,0,0]
( 9)      lt!MDATreq[DR,,,]
(10)       ut?IDISind
(10)       ut?IDISind
( 8)     ut?ICONconf
( 9)      lt!MDATreq[DR,,,]
(10)       ut?IDISind
(10)       ut?IDISind
( 7)    lt!MDATreq[DR,,,]
( 8)     ut?IDISind
( 9)      ut!ICONreq
(10)       lt?MDATind[CR,,,]
( 9)      lt!MDATreq[DR,,,]
(10)       ut?IDISind
(10)       ut?IDISind
( 8)     ut?IDISind
( 9)      lt!MDATreq[DR,,,]
(10)       ut?IDISind
(10)       ut?IDISind
( 7)    lt?MDATind[CR,,,]
( 8)     lt?MDATind[CR,,,]
( 9)      lt?MDATind[CR,,,]
(10)       ut?IDISind
(10)       ut?IDISind
(10)       ut?IDISind
(10)       lt!MDATreq[CC,,,]
(10)       lt!MDATreq[DR,,,]
( 9)      lt!MDATreq[CC,,,]
(10)       ut?ICONconf
(10)       ut?ICONconf
( 9)      lt!MDATreq[DR,,,]
(10)       ut?IDISind
(10)       ut?IDISind
( 8)     lt!MDATreq[CC,,,]
( 9)      ut?ICONconf
(10)       ut!IDATreq[,0,0]
(10)       lt!MDATreq[DR,,,]
( 9)      ut?ICONconf
(10)       lt!MDATreq[DR,,,]
( 8)     lt!MDATreq[DR,,,]
( 9)      ut?IDISind
(10)       ut!ICONreq
(10)       lt!MDATreq[DR,,,]
( 9)      ut?IDISind
(10)       lt!MDATreq[DR,,,]
( 5)    lt!MDATreq[DR,,,]
( 6)     ut?IDISind
( 7)      ut!ICONreq
( 8)       lt?MDATind[CR,,,]
( 9)       lt?MDATind[CR,,,]
(10)        lt?MDATind[CR,,,]
(10)        lt!MDATreq[CC,,,]
(10)        lt!MDATreq[DR,,,]
( 9)       lt!MDATreq[CC,,,]
(10)        ut?ICONconf
(10)        ut?ICONconf
( 9)       lt!MDATreq[DR,,,]
(10)        ut?IDISind
(10)        ut?IDISind
( 9)       lt?MDATind[CR,,,]
(10)        lt?MDATind[CR,,,]
(10)        lt!MDATreq[CC,,,]
(10)        lt!MDATreq[DR,,,]
( 7)      lt!MDATreq[DR,,,]
( 8)       ut?IDISind
( 9)        ut!ICONreq
(10)         lt?MDATind[CR,,,]
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 8)       ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 6)      ut?IDISind
( 7)       lt!MDATreq[DR,,,]
( 8)        ut?IDISind
( 9)         ut!ICONreq
(10)          lt?MDATind[CR,,,]
( 9)         lt!MDATreq[DR,,,]
(10)          ut?IDISind
(10)          ut?IDISind
( 8)        ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)          ut?IDISind
(10)          ut?IDISind
( 7)       lt!MDATreq[DR,,,]
( 8)        ut?IDISind
( 9)         ut!ICONreq
(10)          lt?MDATind[CR,,,]
( 9)         lt!MDATreq[DR,,,]
(10)          ut?IDISind
(10)          ut?IDISind
( 9)         lt?MDATind[CR,,,]
(10)          lt?MDATind[CR,,,]
(10)          lt!MDATreq[CC,,,]
(10)          lt!MDATreq[DR,,,]
( 9)         lt!MDATreq[CC,,,]
(10)          ut?ICONconf
(10)          ut?ICONconf
( 9)         lt!MDATreq[DR,,,]
(10)          ut?IDISind
(10)          ut?IDISind
( 8)        ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)          ut?IDISind
(10)          ut?IDISind
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 6)        ut?IDISind
( 7)         lt!MDATreq[DR,,,]
( 8)          ut?IDISind
( 9)           ut!ICONreq
(10)            lt?MDATind[CR,,,]
( 9)           lt!MDATreq[DR,,,]
(10)            ut?IDISind
(10)            ut?IDISind
( 8)          ut?IDISind
( 9)           lt!MDATreq[DR,,,]
(10)            ut?IDISind
(10)            ut?IDISind
( 7)         lt!MDATreq[DR,,,]
( 8)          ut?IDISind
( 9)           ut!ICONreq
(10)            lt?MDATind[CR,,,]
( 9)           lt!MDATreq[DR,,,]
(10)            ut?IDISind
(10)            ut?IDISind
( 8)          ut?IDISind
( 9)           lt!MDATreq[DR,,,]
(10)            ut?IDISind
(10)            ut?IDISind
( 6)        ut?IDISind
( 7)         lt!MDATreq[DR,,,]
( 8)          ut?IDISind
( 9)           ut!ICONreq
(10)            lt?MDATind[CR,,,]
( 9)           lt!MDATreq[DR,,,]
(10)            ut?IDISind
(10)            ut?IDISind
( 8)          ut?IDISind
( 9)           lt!MDATreq[DR,,,]
(10)            ut?IDISind
(10)            ut?IDISind
( 4)      lt!MDATreq[CC,,,]
( 5)       ut?ICONconf
( 6)        ut!IDATreq[,0,0]
( 7)         lt?MDATind[DT,TRUE,0,0]
( 8)          lt?MDATind[DT,TRUE,0,0]
( 9)           lt?MDATind[DT,TRUE,0,0]
(10)            lt?MDATind[DT,TRUE,0,0]
(10)            lt!MDATreq[AK,TRUE,0,0]
(10)            lt!MDATreq[DR,TRUE,0,0]
( 9)           lt!MDATreq[AK,TRUE,0,0]
(10)            ut!IDATreq[,0,0]
(10)            lt!MDATreq[DR,TRUE,0,0]
( 9)           lt!MDATreq[DR,TRUE,0,0]
(10)            lt?MDATind[DT,TRUE,0,0]
(10)            lt!MDATreq[DR,TRUE,0,0]
( 8)          lt!MDATreq[AK,TRUE,0,0]
( 9)           ut!IDATreq[,0,0]
(10)            lt?MDATind[DT,FALSE,0,0]
( 9)           lt!MDATreq[DR,TRUE,0,0]
(10)            ut?IDISind
(10)            ut?IDISind
```

```
( 8)      lt!MDATreq[DR,TRUE,0,0]
( 9)       lt?MDATind[DT,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 9)       lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 8)      lt?MDATind[DT,TRUE,0,0]
( 9)       lt?MDATind[DT,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[AK,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 9)       lt!MDATreq[AK,TRUE,0,0]
(10)         ut!IDATreq[,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 9)       lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 6)     lt!MDATreq[DR,,,]
( 7)      ut?IDISind
( 8)      ut!ICONreq
( 9)       lt?MDATind[CR,,,]
(10)         lt?MDATind[CR,,,]
(10)         lt!MDATreq[CC,,,]
(10)         lt!MDATreq[DR,,,]
(10)         lt?MDATind[CR,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 7)      ut?IDISind
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 5)     ut?ICONconf
( 6)      lt!MDATreq[DR,,,]
( 7)       ut?IDISind
( 8)       ut!ICONreq
( 9)        lt?MDATind[CR,,,]
(10)          lt?MDATind[CR,,,]
(10)          lt!MDATreq[CC,,,]
(10)          lt!MDATreq[DR,,,]
(10)          lt?MDATind[CR,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 7)      ut?IDISind
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 4)     lt!MDATreq[DR,,,]
( 5)     ut?IDISind
( 6)      ut!ICONreq
( 7)       lt?MDATind[CR,,,]
( 8)       lt?MDATind[CR,,,]
( 9)        lt?MDATind[CR,,,]
(10)          lt?MDATind[CR,,,]
(10)          lt!MDATreq[CC,,,]
(10)          lt!MDATreq[DR,,,]
( 9)        lt!MDATreq[CC,,,]
(10)          ut?ICONconf
(10)          ut?ICONconf
( 9)        lt!MDATreq[DR,,,]
(10)          ut?IDISind
(10)          ut?IDISind
( 8)       lt!MDATreq[CC,,,]
( 9)        ut?ICONconf
(10)          ut!IDATreq[,0,0]
(10)          lt!MDATreq[DR,,,]
( 9)        ut?ICONconf
(10)          lt!MDATreq[DR,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
```

```
( 8)      lt?MDATind[CR,,,]
( 9)       lt?MDATind[CR,,,]
(10)         lt?MDATind[CR,,,]
(10)         lt!MDATreq[CC,,,]
(10)         lt!MDATreq[DR,,,]
( 9)       lt!MDATreq[CC,,,]
(10)         ut?ICONconf
(10)         ut?ICONconf
( 9)       lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 6)     lt!MDATreq[DR,,,]
( 7)      ut?IDISind
( 8)       ut!ICONreq
( 9)        lt?MDATind[CR,,,]
(10)          lt?MDATind[CR,,,]
(10)          lt!MDATreq[CC,,,]
(10)          lt!MDATreq[DR,,,]
(10)          lt?MDATind[CR,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 7)      ut?IDISind
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 5)     ut?IDISind
( 6)      lt!MDATreq[DR,,,]
( 7)       ut?IDISind
( 8)       ut!ICONreq
( 9)        lt?MDATind[CR,,,]
(10)          lt?MDATind[CR,,,]
(10)          lt!MDATreq[CC,,,]
(10)          lt!MDATreq[DR,,,]
(10)          lt?MDATind[CR,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 7)      ut?IDISind
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 1) lt!MDATreq[DR,,,]
( 2)  ut?IDISind
( 3)   ut!ICONreq
( 4)    lt?MDATind[CR,,,]
( 5)     lt?MDATind[CR,,,]
( 6)      lt?MDATind[CR,,,]
( 7)       lt?MDATind[CR,,,]
( 8)        ut?IDISind
( 9)         ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 9)         lt!MDATreq[CC,,,]
(10)           ut!ICONreq
(10)           lt!MDATreq[DR,,,]
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)        ut?IDISind
( 9)         lt!MDATreq[CC,,,]
(10)           ut!ICONreq
(10)           lt!MDATreq[DR,,,]
( 8)        ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)        lt!MDATreq[CC,,,]
( 9)         ut?ICONconf
(10)           ut!IDATreq[,0,0]
(10)           lt!MDATreq[DR,,,]
( 9)         ut?ICONconf
(10)           lt!MDATreq[DR,,,]
( 8)        lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)           ut!ICONreq
(10)           lt!MDATreq[DR,,,]
```

```
( 9)       ut?IDISind
(10)         lt!MDATreq[DR,,,]
( 7)      lt!MDATreq[CC,,,]
( 8)       ut?ICONconf
( 9)        ut!IDATreq[,0,0]
(10)          lt?MDATind[DT,TRUE,0,0]
( 9)        lt!MDATreq[DR,,,]
(10)          ut?IDISind
(10)          ut?IDISind
( 8)       ut?ICONconf
( 9)        lt!MDATreq[DR,,,]
(10)          ut?IDISind
(10)          ut?IDISind
( 7)       lt!MDATreq[DR,,,]
( 8)        ut?IDISind
( 9)         ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)        ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 6)     lt!MDATreq[CC,,,]
( 7)      ut?ICONconf
( 8)       ut!IDATreq[,0,0]
( 9)        lt?MDATind[DT,TRUE,0,0]
(10)          lt?MDATind[DT,TRUE,0,0]
(10)          lt!MDATreq[AK,TRUE,0,0]
(10)          lt!MDATreq[DR,TRUE,0,0]
(10)          lt?MDATind[DT,TRUE,0,0]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 7)      ut?ICONconf
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 6)     lt!MDATreq[DR,,,]
( 7)      ut?IDISind
( 8)       ut!ICONreq
( 9)        lt?MDATind[CR,,,]
(10)          lt?MDATind[CR,,,]
(10)          lt!MDATreq[CC,,,]
(10)          lt!MDATreq[DR,,,]
(10)          lt?MDATind[CR,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 7)      ut?IDISind
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          ut!ICONreq
(10)          lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 5)     lt!MDATreq[CC,,,]
( 6)      ut?ICONconf
( 7)       ut!IDATreq[,0,0]
( 8)        lt?MDATind[DT,TRUE,0,0]
( 9)         lt?MDATind[DT,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[AK,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 9)         lt!MDATreq[AK,TRUE,0,0]
(10)           ut!IDATreq[,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 9)         lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 9)         lt?MDATind[DT,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[AK,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
( 7)       lt!MDATreq[DR,,,]
( 8)        ut?IDISind
( 9)         ut!ICONreq
(10)           lt?MDATind[CR,,,]
```

```
( 9)       lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 8)       ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 6)     ut?ICONconf
( 7)      lt!MDATreq[DR,,,]
( 8)       ut?IDISind
( 9)        ut!ICONreq
(10)          lt?MDATind[CR,,,]
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 8)       ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 5)     lt!MDATreq[DR,,,]
( 6)      ut?IDISind
( 7)       ut!ICONreq
( 8)        lt?MDATind[CR,,,]
( 9)         lt?MDATind[CR,,,]
(10)           lt?MDATind[CR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt!MDATreq[DR,,,]
( 9)         lt!MDATreq[CC,,,]
(10)           ut?ICONconf
(10)           ut?ICONconf
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)        ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 6)     ut?ICONconf
( 7)      lt!MDATreq[DR,,,]
( 8)       ut?IDISind
( 9)        ut!ICONreq
(10)          lt?MDATind[CR,,,]
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 8)       ut?IDISind
( 9)        lt!MDATreq[CC,,,]
(10)         ut!ICONreq
(10)         lt!MDATreq[DR,,,]
( 8)       ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 8)       lt!MDATreq[CC,,,]
( 9)        ut?ICONconf
(10)          ut!IDATreq[,0,0]
(10)          lt!MDATreq[DR,,,]
( 9)        ut?ICONconf
(10)          lt!MDATreq[DR,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
```

```
(10)        ut!ICONreq
(10)         lt!MDATreq[DR,,,]
( 9)       ut?IDISind
(10)         lt!MDATreq[DR,,,]
( 7)      lt!MDATreq[CC,,,]
( 8)       ut?ICONconf
( 9)       ut!IDATreq[,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 9)       lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 8)       ut?ICONconf
( 9)       lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 7)      lt!MDATreq[DR,,,]
( 8)       ut?IDISind
( 9)       ut!ICONreq
(10)         lt?MDATind[CR,,,]
( 9)       lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 8)       ut?IDISind
( 9)       lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 6)      lt!MDATreq[CC,,,]
( 7)       ut?ICONconf
( 8)       ut!IDATreq[,0,0]
( 9)       lt?MDATind[DT,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[AK,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 8)       lt!MDATreq[DR,,,]
( 9)       ut?IDISind
(10)         ut!ICONreq
(10)         lt!MDATreq[DR,,,]
( 9)       ut?IDISind
(10)         lt!MDATreq[DR,,,]
( 7)      ut?ICONconf
( 8)       lt!MDATreq[DR,,,]
( 9)       ut?IDISind
(10)         ut!ICONreq
(10)         lt!MDATreq[DR,,,]
( 9)       ut?IDISind
(10)         lt!MDATreq[DR,,,]
( 6)      lt!MDATreq[DR,,,]
( 7)       ut?IDISind
( 8)       ut!ICONreq
( 9)       lt?MDATind[CR,,,]
(10)         lt?MDATind[CR,,,]
(10)         lt!MDATreq[CC,,,]
(10)         lt!MDATreq[DR,,,]
(10)         lt?MDATind[CR,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)       ut?IDISind
(10)         ut!ICONreq
(10)         lt!MDATreq[DR,,,]
( 9)       ut?IDISind
(10)         lt!MDATreq[DR,,,]
( 3)    lt!MDATreq[DR,,,]
( 4)     ut?IDISind
( 5)      ut!ICONreq
( 6)       lt?MDATind[CR,,,]
( 7)       lt?MDATind[CR,,,]
( 8)       lt?MDATind[CR,,,]
( 9)       lt?MDATind[CR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
(10)         ut?IDISind
(10)         lt!MDATreq[CC,,,]
(10)         lt!MDATreq[DR,,,]
( 9)       lt!MDATreq[CC,,,]
(10)         ut?ICONconf
(10)         ut?ICONconf
( 9)       lt!MDATreq[DR,,,]
(10)         ut?IDISind
(10)         ut?IDISind
( 8)       lt!MDATreq[CC,,,]
( 9)       ut?ICONconf
(10)        ut!IDATreq[,0,0]
(10)         lt!MDATreq[DR,,,]
( 9)       ut?ICONconf
(10)         lt!MDATreq[DR,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)       ut?IDISind
(10)         ut!ICONreq
(10)         lt!MDATreq[DR,,,]
( 9)       ut?IDISind
(10)         lt!MDATreq[DR,,,]
( 5)      lt!MDATreq[DR,,,]
( 6)       ut?IDISind
( 7)       ut!ICONreq
( 8)        lt?MDATind[CR,,,]
( 9)         lt?MDATind[CR,,,]
(10)           lt?MDATind[CR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt!MDATreq[DR,,,]
( 9)         lt!MDATreq[CC,,,]
(10)           ut?ICONconf
(10)           ut?ICONconf
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         lt!MDATreq[CC,,,]
( 9)         ut?ICONconf
(10)           ut!IDATreq[,0,0]
(10)           lt!MDATreq[DR,,,]
( 9)         ut?ICONconf
(10)           lt!MDATreq[DR,,,]
( 8)         lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)           ut!ICONreq
(10)           lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)           lt!MDATreq[DR,,,]
( 7)        lt!MDATreq[CC,,,]
( 8)         ut?IDISind
( 9)         ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 6)        ut?IDISind
( 7)         lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)         ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)         ut!ICONreq
(10)           lt?MDATind[CR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt!MDATreq[DR,,,]
( 9)         lt!MDATreq[CC,,,]
(10)           ut?ICONconf
(10)           ut?ICONconf
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
(10)           ut?IDISind
( 6)        ut?IDISind
( 7)         ut!ICONreq
( 8)          lt?MDATind[CR,,,]
( 9)          lt?MDATind[CR,,,]
(10)            lt?MDATind[CR,,,]
(10)            lt!MDATreq[CC,,,]
(10)            lt!MDATreq[DR,,,]
( 9)          lt!MDATreq[CC,,,]
(10)            ut?ICONconf
(10)            ut?ICONconf
( 9)          lt!MDATreq[DR,,,]
(10)            ut?IDISind
(10)            ut?IDISind
( 8)          ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)            ut?IDISind
(10)            ut?IDISind
( 6)         ut?IDISind
( 7)          lt!MDATreq[DR,,,]
( 8)          ut?IDISind
( 9)          ut!ICONreq
(10)            lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)            ut?IDISind
(10)            ut?IDISind
( 8)          ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)            ut?IDISind
(10)            ut?IDISind
( 7)         lt!MDATreq[CC,,,]
( 8)          ut?ICONconf
( 9)          ut!IDATreq[,0,0]
(10)            lt?MDATind[DT,TRUE,0,0]
```

```
( 7)    ut!ICONreq
( 8)     lt?MDATind[CR,,,]
( 9)     lt?MDATind[CR,,,]
(10)      lt?MDATind[CR,,,]
(10)      lt!MDATreq[CC,,,]
(10)      lt!MDATreq[DR,,,]
( 9)     lt!MDATreq[CC,,,]
(10)      ut?ICONconf
(10)      ut?ICONconf
( 9)     lt!MDATreq[DR,,,]
```

```
(10)      ut?IDISind
(10)      ut?IDISind
( 9)     lt?MDATind[CR,,,]
(10)      lt?MDATind[CR,,,]
(10)      lt!MDATreq[CC,,,]
(10)      lt!MDATreq[DR,,,]
( 7)     lt!MDATreq[DR,,,]
( 8)     ut?IDISind
( 9)     ut!ICONreq
(10)      lt?MDATind[CR,,,]
```

```
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 8)    ut?IDISind
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 6)   ut?IDISind
( 7)    lt!MDATreq[DR,,,]
( 8)    ut?IDISind
```

```
( 9)     ut!ICONreq
(10)      lt?MDATind[CR,,,]
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
( 8)    ut?IDISind
( 9)     lt!MDATreq[DR,,,]
(10)      ut?IDISind
(10)      ut?IDISind
```

End of generation: 92-05-12 21:35:16
End of Testcases

B.2 Datenphase für die Testkonfiguration 1

TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 4
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : ENV, System-Process : Empfaenger
Signalroute : ISAP
REDUCED SIGNALS:
Process: Empfaenger, State: warten, Signal: IDISreq
Process: Empfaenger, State: verbunden, Signal: IDISreq
Process: Empfaenger, State: unterbrochen, Signal: IDISreq
Process: Empfaenger, State: verbunden1, Signal: IDISreq
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-12 17:46:27

```
( 1) ut!IDATreq[,0,0]
( 2)  lt?IDATind[0,0]
( 3)  ut?IDISind
( 4)   ut!ICONreq
( 4)   ut!ICONreq
( 3)  ut?IDISind
( 4)   ut!ICONreq
( 3)  ut!IDATreq[,0,0]
( 4)   lt?IDATind[0,0]
```

```
( 4)   lt?IDATind[0,0]
( 3)  ut!IDATreq[,0,0]
( 4)   lt?IDATind[0,0]
( 4)   lt?IDATind[0,0]
( 3)  ut!IDATreq[,0,0]
( 4)   lt?IDATind[0,0]
( 4)   lt?IDATind[0,0]
( 3)  ut!IDATreq[,0,0]
( 4)   lt?IDATind[0,0]
```

```
( 4)   lt?IDATind[0,0]
( 2)  lt?IDATind[0,0]
( 3)  ut?IDISind
( 4)   ut!ICONreq
( 4)   ut!ICONreq
( 3)  ut?IDISind
( 4)   ut!ICONreq
( 3)  ut!IDATreq[,0,0]
( 4)   lt?IDATind[0,0]
```

```
( 4)   lt?IDATind[0,0]
( 3)  ut!IDATreq[,0,0]
( 4)   lt?IDATind[0,0]
( 4)   lt?IDATind[0,0]
( 3)  ut!IDATreq[,0,0]
( 4)   lt?IDATind[0,0]
( 4)   lt?IDATind[0,0]
```

End of generation: 92-05-12 17:49:04
End of Testcases

B.3 Datenphase für die Testkonfiguration 3

TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 11
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REDUCED SIGNALS:
Process: Empfaenger, State: warten, Signal: IDISreq
Process: Empfaenger, State: unterbrochen, Signal: IDISreq
Process: Empfaenger, State: verbunden, Signal: IDISreq
Process: Empfaenger, State: verbunden1, Signal: IDISreq

REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-14 15:31:28

```
( 1) ut!IDATreq[,0,0]
( 2)  lt?MDATind[DT,TRUE,0,0]
( 3)  lt?MDATind[DT,TRUE,0,0]
( 4)  lt?MDATind[DT,TRUE,0,0]
( 5)  lt?MDATind[DT,TRUE,0,0]
( 6)   ut?IDISind
( 7)   ut!ICONreq
( 8)    lt?MDATind[CR,TRUE,0,0]
( 9)    lt?MDATind[CR,TRUE,0,0]
(10)     lt?MDATind[CR,TRUE,0,0]
(11)     lt?MDATind[CR,TRUE,0,0]
(11)     lt!MDATreq[CC,TRUE,0,0]
(10)     lt!MDATreq[CC,TRUE,0,0]
(11)      ut?ICONconf
( 9)    lt!MDATreq[CC,TRUE,0,0]
(10)     ut?ICONconf
(11)      ut!IDATreq[,0,0]
( 9)    lt?MDATind[CR,TRUE,0,0]
(10)     lt?MDATind[CR,TRUE,0,0]
(11)     lt?MDATind[CR,TRUE,0,0]
(11)     lt!MDATreq[CC,TRUE,0,0]
(10)     lt!MDATreq[CC,TRUE,0,0]
(11)      ut?ICONconf
( 7)   lt!MDATreq[AK,TRUE,0,0]
( 8)    ut!ICONreq
( 9)     lt?MDATind[CR,TRUE,0,0]
(10)     lt?MDATind[CR,TRUE,0,0]
(11)     lt?MDATind[CR,TRUE,0,0]
(11)     lt!MDATreq[CC,TRUE,0,0]
(10)     lt!MDATreq[CC,TRUE,0,0]
(11)      ut?ICONconf
(10)     lt?MDATind[CR,TRUE,0,0]
(11)     lt?MDATind[CR,TRUE,0,0]
(11)     lt!MDATreq[CC,TRUE,0,0]
( 6)   ut?IDISind
( 7)   lt!MDATreq[AK,TRUE,0,0]
( 8)    ut!ICONreq
( 9)     lt?MDATind[CR,TRUE,0,0]
(10)     lt?MDATind[CR,TRUE,0,0]
(11)     lt?MDATind[CR,TRUE,0,0]
(11)     lt!MDATreq[CC,TRUE,0,0]
(10)     lt!MDATreq[CC,TRUE,0,0]
(11)      ut?ICONconf
(10)     lt?MDATind[CR,TRUE,0,0]
(11)     lt?MDATind[CR,TRUE,0,0]
(11)     lt!MDATreq[CC,TRUE,0,0]
( 6)   lt!MDATreq[AK,TRUE,0,0]
( 7)    ut!IDATreq[,0,0]
( 8)     lt?MDATind[DT,FALSE,0,0]
( 9)     lt?MDATind[DT,FALSE,0,0]
(10)      lt?MDATind[DT,FALSE,0,0]
(11)      lt?MDATind[DT,FALSE,0,0]
(11)      lt!MDATreq[AK,FALSE,0,0]
(10)      lt!MDATreq[AK,FALSE,0,0]
(11)       ut!IDATreq[,0,0]
( 9)     lt!MDATreq[AK,FALSE,0,0]
(10)      ut!IDATreq[,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
( 9)     lt?MDATind[DT,FALSE,0,0]
(10)      lt?MDATind[DT,FALSE,0,0]
(11)      lt?MDATind[DT,FALSE,0,0]
(11)      lt!MDATreq[AK,FALSE,0,0]
(10)      lt!MDATreq[AK,FALSE,0,0]
(11)       ut!IDATreq[,0,0]
( 5)   lt!MDATreq[AK,TRUE,0,0]
( 6)    ut!IDATreq[,0,0]
( 7)     lt?MDATind[DT,FALSE,0,0]
( 8)     lt?MDATind[DT,FALSE,0,0]
( 9)      lt?MDATind[DT,FALSE,0,0]
(10)      lt?MDATind[DT,FALSE,0,0]
(11)       ut?IDISind
(11)       ut?IDISind
(11)       lt!MDATreq[AK,FALSE,0,0]
(10)       lt!MDATreq[AK,FALSE,0,0]
(11)       ut!IDATreq[,0,0]
( 9)      lt!MDATreq[AK,FALSE,0,0]
(10)       ut!IDATreq[,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
( 8)     lt!MDATreq[AK,FALSE,0,0]
( 9)       ut!IDATreq[,0,0]
(10)       lt?MDATind[DT,TRUE,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
(11)       lt!MDATreq[AK,TRUE,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
( 9)       ut!IDATreq[,0,0]
(10)       lt?MDATind[DT,TRUE,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
(11)       lt!MDATreq[AK,TRUE,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
( 7)     lt?MDATind[DT,FALSE,0,0]
( 8)      lt?MDATind[DT,FALSE,0,0]
( 9)       lt?MDATind[DT,FALSE,0,0]
(10)       ut?IDISind
(11)       ut!ICONreq
(11)       lt?MDATind[AK,FALSE,0,0]
(10)       ut?IDISind
(11)       lt!MDATreq[AK,FALSE,0,0]
(10)       lt!MDATreq[AK,FALSE,0,0]
(11)       ut!IDATreq[,0,0]
( 9)      lt!MDATreq[AK,FALSE,0,0]
(10)       ut!IDATreq[,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
( 8)     lt!MDATreq[AK,FALSE,0,0]
( 9)       ut!IDATreq[,0,0]
(10)       lt?MDATind[DT,TRUE,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
(11)       lt!MDATreq[AK,TRUE,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
( 3)   lt!MDATreq[AK,TRUE,0,0]
( 4)    ut!IDATreq[,0,0]
( 5)     lt?MDATind[DT,FALSE,0,0]
( 6)      lt?MDATind[DT,FALSE,0,0]
( 7)      lt?MDATind[DT,FALSE,0,0]
( 8)      lt?MDATind[DT,FALSE,0,0]
( 9)       ut?IDISind
(10)       ut!ICONreq
(11)       lt?MDATind[CR,FALSE,0,0]
(10)       lt!MDATreq[AK,FALSE,0,0]
(11)       ut!ICONreq
( 9)     ut?IDISind
(10)      lt!MDATreq[AK,FALSE,0,0]
(11)      ut!ICONreq
( 9)     lt!MDATreq[AK,FALSE,0,0]
(10)      ut!IDATreq[,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
( 8)     lt!MDATreq[AK,FALSE,0,0]
( 9)      ut!IDATreq[,0,0]
(10)      lt?MDATind[DT,TRUE,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
(11)      lt!MDATreq[AK,TRUE,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
( 7)    lt!MDATreq[AK,FALSE,0,0]
( 8)     ut!IDATreq[,0,0]
( 9)      lt?MDATind[DT,TRUE,0,0]
(10)      lt?MDATind[DT,TRUE,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
(11)      lt!MDATreq[AK,TRUE,0,0]
(10)      lt!MDATreq[AK,TRUE,0,0]
(11)       ut!IDATreq[,0,0]
( 6)    lt!MDATreq[AK,FALSE,0,0]
( 7)     ut!IDATreq[,0,0]
( 8)      lt?MDATind[DT,TRUE,0,0]
( 9)      lt?MDATind[DT,TRUE,0,0]
(10)      lt?MDATind[DT,TRUE,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
(11)      lt!MDATreq[AK,TRUE,0,0]
(10)      lt!MDATreq[AK,TRUE,0,0]
(11)       ut!IDATreq[,0,0]
( 9)      lt!MDATreq[AK,TRUE,0,0]
(10)       ut!IDATreq[,0,0]
(11)       lt?MDATind[DT,FALSE,0,0]
( 9)      lt?MDATind[DT,TRUE,0,0]
(10)      lt?MDATind[DT,TRUE,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
(11)      lt!MDATreq[AK,TRUE,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
( 8)     lt!MDATreq[AK,FALSE,0,0]
( 9)      ut!IDATreq[,0,0]
(10)      lt?MDATind[DT,TRUE,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
(11)      lt!MDATreq[AK,TRUE,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
( 7)    lt!MDATreq[AK,FALSE,0,0]
( 8)     ut!IDATreq[,0,0]
( 9)      lt?MDATind[DT,TRUE,0,0]
(10)      lt?MDATind[DT,TRUE,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
(11)      lt!MDATreq[AK,TRUE,0,0]
(10)      lt!MDATreq[AK,TRUE,0,0]
(11)       ut!IDATreq[,0,0]
(10)      lt?MDATind[DT,TRUE,0,0]
(11)      lt?MDATind[DT,TRUE,0,0]
(11)      lt!MDATreq[AK,TRUE,0,0]
( 3)   lt?MDATind[DT,TRUE,0,0]
( 4)   lt?MDATind[DT,TRUE,0,0]
( 5)    lt?MDATind[DT,TRUE,0,0]
( 6)     ut?IDISind
( 7)     ut!ICONreq
( 8)      lt?MDATind[CR,TRUE,0,0]
( 9)      lt?MDATind[CR,TRUE,0,0]
(10)       lt?MDATind[CR,TRUE,0,0]
(11)       lt?MDATind[CR,TRUE,0,0]
(11)       lt!MDATreq[CC,TRUE,0,0]
(10)       lt!MDATreq[CC,TRUE,0,0]
(11)        ut?ICONconf
( 9)      lt!MDATreq[CC,TRUE,0,0]
(10)       ut?ICONconf
(11)        ut!IDATreq[,0,0]
( 9)      lt?MDATind[CR,TRUE,0,0]
(10)       lt?MDATind[CR,TRUE,0,0]
(11)       lt?MDATind[CR,TRUE,0,0]
(11)       lt!MDATreq[CC,TRUE,0,0]
(10)       lt!MDATreq[CC,TRUE,0,0]
(11)        ut?ICONconf
( 7)     lt!MDATreq[AK,TRUE,0,0]
( 8)      ut!ICONreq
( 9)       lt?MDATind[CR,TRUE,0,0]
(10)       lt?MDATind[CR,TRUE,0,0]
(11)       lt?MDATind[CR,TRUE,0,0]
(11)       lt!MDATreq[CC,TRUE,0,0]
(10)       lt!MDATreq[CC,TRUE,0,0]
(11)        ut?ICONconf
(10)       lt?MDATind[CR,TRUE,0,0]
(11)       lt?MDATind[CR,TRUE,0,0]
(11)       lt!MDATreq[CC,TRUE,0,0]
( 6)    ut?IDISind
( 7)    lt!MDATreq[AK,TRUE,0,0]
( 8)     ut!ICONreq
( 9)      lt?MDATind[CR,TRUE,0,0]
(10)      lt?MDATind[CR,TRUE,0,0]
(11)      lt?MDATind[CR,TRUE,0,0]
(11)      lt!MDATreq[CC,TRUE,0,0]
(10)      lt!MDATreq[CC,TRUE,0,0]
(11)       ut?ICONconf
(10)      lt?MDATind[CR,TRUE,0,0]
(11)      lt?MDATind[CR,TRUE,0,0]
(11)      lt!MDATreq[CC,TRUE,0,0]
( 6)    lt!MDATreq[AK,TRUE,0,0]
( 7)     ut!IDATreq[,0,0]
( 8)      lt?MDATind[DT,FALSE,0,0]
( 9)      lt?MDATind[DT,FALSE,0,0]
(10)       lt?MDATind[DT,FALSE,0,0]
(11)       lt?MDATind[DT,FALSE,0,0]
(11)       lt!MDATreq[AK,FALSE,0,0]
(10)       lt!MDATreq[AK,FALSE,0,0]
(11)        ut!IDATreq[,0,0]
( 9)      lt!MDATreq[AK,FALSE,0,0]
(10)       ut!IDATreq[,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
( 9)      lt?MDATind[DT,FALSE,0,0]
(10)       lt?MDATind[DT,FALSE,0,0]
(11)       lt?MDATind[DT,FALSE,0,0]
(11)       lt!MDATreq[AK,FALSE,0,0]
(10)       lt!MDATreq[AK,FALSE,0,0]
(11)        ut!IDATreq[,0,0]
( 5)    lt!MDATreq[AK,TRUE,0,0]
( 6)     ut!IDATreq[,0,0]
( 7)      lt?MDATind[DT,FALSE,0,0]
( 8)      lt?MDATind[DT,FALSE,0,0]
( 9)       lt?MDATind[DT,FALSE,0,0]
(10)       lt?MDATind[DT,FALSE,0,0]
(11)        ut?IDISind
(11)        ut?IDISind
(11)        lt!MDATreq[AK,FALSE,0,0]
(10)        lt!MDATreq[AK,FALSE,0,0]
(11)        ut!IDATreq[,0,0]
( 9)       lt!MDATreq[AK,FALSE,0,0]
(10)        ut!IDATreq[,0,0]
(11)        lt?MDATind[DT,TRUE,0,0]
( 8)      lt!MDATreq[AK,FALSE,0,0]
( 9)       ut!IDATreq[,0,0]
(10)       lt?MDATind[DT,TRUE,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
(11)       lt!MDATreq[AK,TRUE,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
( 8)      lt?MDATind[DT,FALSE,0,0]
( 9)       lt?MDATind[DT,FALSE,0,0]
(10)        ut?IDISind
```

```
(11)        ut?IDISind
(11)        lt!MDATreq[AK,FALSE,0,0]
(10)        lt!MDATreq[AK,FALSE,0,0]
(11)        ut!IDATreq[,0,0]
( 9)      lt!MDATreq[AK,FALSE,0,0]
(10)        ut!IDATreq[,0,0]
(11)        lt?MDATind[DT,TRUE,0,0]
( 4)    lt!MDATreq[AK,TRUE,0,0]
( 5)    ut!IDATreq[,0,0]
( 6)     lt?MDATind[DT,FALSE,0,0]
( 7)     lt?MDATind[DT,FALSE,0,0]
( 8)     lt?MDATind[DT,FALSE,0,0]
( 9)     lt?MDATind[DT,FALSE,0,0]
(10)        ut?IDISind
(11)        ut!ICONreq
```

```
(11)        lt!MDATreq[AK,FALSE,0,0]
(10)        ut?IDISind
(11)        lt!MDATreq[AK,FALSE,0,0]
(10)        lt!MDATreq[AK,FALSE,0,0]
(11)        ut!IDATreq[,0,0]
( 9)      lt!MDATreq[AK,FALSE,0,0]
(10)        ut!IDATreq[,0,0]
(11)        lt?MDATind[DT,TRUE,0,0]
( 8)     lt!MDATreq[AK,FALSE,0,0]
( 9)      ut!IDATreq[,0,0]
(10)        lt?MDATind[DT,TRUE,0,0]
(11)        lt?MDATind[DT,TRUE,0,0]
(11)        lt!MDATreq[AK,TRUE,0,0]
(11)        lt?MDATind[DT,TRUE,0,0]
( 7)    lt!MDATreq[AK,FALSE,0,0]
```

```
( 8)      ut!IDATreq[,0,0]
( 9)       lt?MDATind[DT,TRUE,0,0]
(10)       lt?MDATind[DT,TRUE,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
(11)       lt!MDATreq[AK,TRUE,0,0]
(10)       lt!MDATreq[AK,TRUE,0,0]
(11)       ut!IDATreq[,0,0]
(10)       lt?MDATind[DT,TRUE,0,0]
(11)       lt?MDATind[DT,TRUE,0,0]
(11)       lt!MDATreq[AK,TRUE,0,0]
( 7)     lt?MDATind[DT,FALSE,0,0]
( 8)     lt?MDATind[DT,FALSE,0,0]
( 9)      lt?MDATind[DT,FALSE,0,0]
(10)       ut?IDISind
(11)       ut!ICONreq
```

```
(11)        lt!MDATreq[AK,FALSE,0,0]
(10)        ut?IDISind
(11)        lt!MDATreq[AK,FALSE,0,0]
(10)        lt!MDATreq[AK,FALSE,0,0]
(11)        ut!IDATreq[,0,0]
( 9)      lt!MDATreq[AK,FALSE,0,0]
(10)        ut!IDATreq[,0,0]
(11)        lt?MDATind[DT,TRUE,0,0]
( 8)     lt!MDATreq[AK,FALSE,0,0]
( 9)      ut!IDATreq[,0,0]
(10)        lt?MDATind[DT,TRUE,0,0]
(11)        lt?MDATind[DT,TRUE,0,0]
(11)        lt!MDATreq[AK,TRUE,0,0]
(11)        lt?MDATind[DT,TRUE,0,0]
```

End of generation: 92-05-14 16:18:09
End of Testcases

B.4 Verbindungsabbau für die Testkonfiguration 1

TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 2
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : ENV, System-Process : Empfaenger
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-10 12:26:17

```
( 1) ut!IDATreq[,0,0]
( 2)  lt?IDATind[0,0]
( 1) lt!IDISreq
( 2)  ut?IDISind
( 2)  lt!IDISreq
```

End of generation: 92-05-10 12:26:30
End of Testcases

B.5 Verbindungsabbau für die Testkonfiguration 2

TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 2
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-10 12:28:16

```
( 1) ut!IDATreq[,0,0]
( 2)  lt?MDATind[DT,TRUE,0,0]
( 1) lt!MDATreq[DR,,,]
( 2)  ut?IDISind
```

```
( 2)  ut?IDISind
( 2)  ut?IDISind
( 2)  ut?IDISind
( 1) lt!MDATreq[CC,,,]
( 2)  ut!IDATreq[,0,0]
( 2)  lt!MDATreq[DR,,,]
( 2)  lt!MDATreq[CC,,,]
( 2)  lt!MDATreq[AK,FALSE,,]
( 1) lt!MDATreq[AK,FALSE,,]
( 2)  ut!IDATreq[,0,0]
( 2)  lt!MDATreq[DR,FALSE,,]
( 2)  lt!MDATreq[CC,FALSE,,]
( 2)  lt!MDATreq[AK,FALSE,,]
```

End of generation: 92-05-10 12:28:39
End of Testcases

B.6 Verbindungsabbau für die Testkonfiguration 3

```
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE :          2
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-12 19:28:41

( 1) ut!IDATreq[,0,0]
( 2)  lt?MDATind[DT,TRUE,0,0]
( 1) lt!MDATreq[DR,,,]
( 2)  ut?IDISind
( 2)  ut?IDISind
```

End of generation: 92-05-12 19:28:51
End of Testcases

Anhang C Dialog und Ausgabe REDUC-1 und REDUC-2

C.1 Dialog mit REDUC-1 (analog für REDUC-2)

Eingabe:

- Eingabedatei (evtl. mit Pfad)
- Ausgabedatei (evtl. mit Pfad)

Bildschirmanzeige während der Verarbeitung:

- Anzahl eingelesener Zeilen

Ausgabe:

- am Bildschirm: Angaben zur Statistik
- Ausgabedatei

C.2: Beispiel einer reduzierten Testdatendatei

Jeder reduzierten Datei stellen die Algorithmen REDUC-1/2 eine Zeile zur Kennzeichnung voran. Am Ende der reduzierten Datei beinhalten zusätzlich angehängte Zeilen Angaben über die benötigte Zeit, Anzahl der verarbeiteten und ausgegebenen Zeilen und den Reduktionsfaktor.

```
REDUC-1: This is REDUCED version of file INRES.3
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE :         3
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : ENV, System-Process : Empfaenger
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-03 14:35:18
( 1) lt!IDISreq
( 2)  lt!IDISreq
( 3)   ut?IDISind
( 2)  ut?IDISind
( 3)   lt!IDISreq
( 3)   ut!ICONreq
( 1) ut!ICONreq
( 2)  lt?ICONind
( 3)   lt!IDISreq
( 3)   lt!ICONresp
( 3)   ut?IDISind
End of generation: 92-05-03 14:36:38
End of Testcases@
Tree reduction ended : 06-03-93 11:55:26
Reduction time      : 00:00:00
Reduction factor    : 63% (29 -> 11)
Count of paths      : 6
```

Anhang D LISP - Implementation REDUC-1/2

D.1 Algorithmus REDUC-1

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; LISP-Algorithmus zur Reduktion eines Tracebaumes
; - Eingabe : Tracebaum              (Textfile)
; - Ausgabe : Korrigierter Tracebaum  (Textfile)
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(in-package "USER")
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Globale Variablen
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defvar *traceline "Start")       ; vollst. gelesene Zeile       (sequence)
(defvar *tracelevel 0)            ; Tiefe des Traces             (numeric)
(defvar *tracetag "None")         ; Trace-Tag                    (sequence)
(defvar *result nil)              ; Resultierender Trace-Baum    (list)
(defvar *lncnt 0)                 ; Anzahl eingelesener Zeilen   (numeric)
(defvar *lncnt2 0)                ; Anzahl ausgegebener Zeilen   (numeric)
(defvar *pathcnt 0)               ; Anzahl ausgegebener Pfade    (numeric)
(defvar *instream nil)            ; Eingabe-Handle               (stream)
(defvar *outstream nil)           ; Ausgabe-Handle               (stream)
(defvar *read-error nil)          ; Lese-Fehler-Flag             (boolean)
(defvar *infilename "")           ; Name Eingabe-Datei           (sequence)
(defvar *outfilename "")          ; Name Ausgabe-Datei           (sequence)
(defvar *time-elapsed 0)          ; Zeitaufwand Reduktion        (numeric)
;
(defun REDUC ()
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Verschiedene kleine Hilfsfunktionen
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun CLOCK (&optional universal-time)
  "Return a string telling current time."
  (multiple-value-bind (sec min hrs)
    (decode-universal-time (if (null universal-time)
        (get-universal-time) universal-time))
    (format nil "~2,'0D:~2,'0D:~2,'0D"
            (if universal-time (1- hrs) hrs) min sec)
) )

(defun DATE ()
  "Return a string telling current date."
  (multiple-value-bind (sec min hrs date month year)
    (decode-universal-time (get-universal-time))
    (declare (ignore sec min hrs))
    (format nil "~2,'0D-~2,'0D-~2,'0D"
        date month (mod year 100))
) ) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Bestimmen der Input- und Outputfiles
;--------------------------------------------------------------------------
; Parameter                : keine
; Resultat                 : T = Ok / NIL = Fehler
; Veraenderte glob. Variabeln : *infilename, *outfilename
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun SELECTFILES ()
  (loop                                ; Schleife fuer Eingabefehler
    (terpri) (princ "Input file  : ")
    (setq *infilename (read-line))
    (when (equal *infilename "")            ; Wenn keine Eingabe
        (return-from selectfiles nil))      ;  dann Abbruch
```

```
   (when (probe-file *infilename)              ; Wenn Datei existiert,
        (return))                              ;  dann beenden der Schleife
   (terpri)                                    ;  sonst Fehlermeldung
   (princ "Error: input file does not exist !")
  )
  (loop                                        ; Schleife fuer Eingabefehler
   (terpri) (princ "Output file : ")
   (setq *outfilename (read-line))
   (cond
    ((equal *outfilename "")                   ; Abbrechen ?
     (return-from selectfiles nil))
    ((probe-file *outfilename)                 ; Datei existiert ?
     (princ "Replacing existing output file (y/N) : ")
     (when (equalp (read-line) "y")            ; Datei ueberschreiben ?
       (return T)))
    (T (return T))                             ; Datei neu, fertig
) ) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Oeffnen der Eingabe- & Ausgabedatei
;---------------------------------------------------------------------------
; Parameter                  : keine
; Resultat                   : T = Ok / NIL = Fehler
; Veraenderte glob. Variabeln : *instream, *traceline
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun OPENFILES ()
  (setq *outstream                             ; Setzen des Ausgabestroms
       (open *outfilename :direction :output   ; auf die Ausgabedatei
            :if-exists :supersede))
  (format t "~%~A~%" "Reading input file ...")
  (setq *instream (open *infilename :direction :input)) ; Oeffnen der Datei
  (setq *traceline (read-line *instream nil "End of File"))
  (when (string-equal *traceline "REDUC" :end1 5)
    (write-line "Input file is already a reduced version !")
    (princ "Continue anyway [y/N] : ")
    (when (not (equalp (read-line) "y"))
     (write-line "Program aborted.")
     (return-from reduc nil)
  ) )
  (format *outstream "~A~:@(~A~)~%"             ; Ausgabedatei markieren
        "REDUC-1: This is a REDUCED version of file "
        *infilename)
  (write-line *traceline *outstream)
  (loop                                        ; Einlese-Schleife
   (setq *traceline (read-line *instream       ; Zeile lesen solange
                    nil "End of File"))        ; Dateiende nicht erreicht
   (write-line *traceline *outstream)
   (cond
    ((< (length *traceline) 5))
    ((string-equal *traceline "start" :end1 5) ; Wenn erster Trace gefunden
     (return T))                               ;  dann beende Schleife
    ((string-equal *traceline "end" :end1 3)   ; Wenn Dateiende erreicht
     (terpri)
     (princ "Eror: No trace lines found !")    ; -> Fehlermeldung
     (return-from openfiles nil)               ;  und Abbruch
) ) ) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Einlesen einer Textzeile und Parsen nach Trace-Level und Trace-Tag
;---------------------------------------------------------------------------
; Parameter                  : keine
; Resultat                   : ohne Bedeutung
; Veraenderte glob. Variabeln : *traceline, *tracelevel, *tracetag
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun GETTRACELINE ()
  (setq *traceline (read-line *instream nil "*EOF"))
  (cond
   ((< (length *traceline) 4)
    (write-line "Reading error: ignoring invalid line"))
```

```
   ((string-equal *traceline "End" :end1 3)
     (format t "~A~A~%" #\Return
             "End of input file found")
     (setq *read-error T))
   ((string-equal *traceline "*EOF" :end1 4)
     (format t "~A~A~D~%" #\Return "line : " *lncnt)
     (write-line " Reading error: unexpected end of file !")
     (setq *read-error T))
   (T
     (setq *tracetag (string-trim " " (subseq *traceline 5)))
     (setq *tracelevel (car (read-from-string *traceline t nil)))
     (setq *lncnt (1+ *lncnt))
     (when (= (mod *lncnt 250) 0)
       (format t "~A~D" #\Return *lncnt)
  ) ) )
  (when *read-error
    (setq *tracetag "")
    (setq *tracelevel 0)
) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Rekursive Funktion:
; Einlesen der Trace-Zeilen mit gleichzeitigem Aufbau des Trace-Baumes
;---------------------------------------------------------------------------
; Parameter                   : bisheriger Unterbaum, aktueller Tracelevel
; Resultat                    : neuer/korrigierter Unterbaum
; Veraenderte glob. Variabeln :  [*traceline, *tracelevel, *tracetag]
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun BUILDTREE (subtree actlevel)
  (let ((localtag *tracetag) (treearg nil))      ; lokale Var. definieren
    (loop                                     ; Rekursions-Schleife
      (if (> actlevel *tracelevel) (return))     ; Ende der Rekursion
      (if (= actlevel *tracelevel)              ; Verarb. auf dieser Ebene ?
      (progn                                   ; Wenn ja, dann ...
        (when (not (assoc *tracetag             ; Blatt/Ast NICHT schon
                  subtree                      ;  vorhanden ?
                  :test #'equal))
;             (if (equal subtree '(nil))        ; Ast leer ?
;               (setq subtree nil))
           (setq subtree (acons *tracetag '(nil) ; ja -> einfuegen
                        subtree))
        )
        (setq localtag *tracetag)               ; Merke akt. Blatt/Ast
        (GETTRACELINE)
      )
      (progn                                   ; Wenn nein, dann ...
        (setq treearg (cadr (assoc localtag      ; Unterbaum des aktuellen
                           subtree            ;  Astes bestimmen
                           :test #'equal)))
        (setq treearg (BUILDTREE treearg         ; Rekursions-Aufruf
                        (1+ actlevel)))
        (setq subtree (subst (cons localtag (list treearg))
                      (assoc localtag         ; Resultierender Unterbaum
                           subtree           ;  in 'subtree' ersetzen
                           :test #'equal)
                      subtree))
    ) ) )
    subtree                                   ; Unterbaum zurueckgeben
) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Ausgabe des resultierenden Trace-Baumes
;---------------------------------------------------------------------------
; Parameter                   : Trace-Baum, Start-Trace-Level
; Resultat                    : ohne Bedeutung
; Veraenderte glob. Variabeln :  -
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun SHOWTREE (stree level)
  (let (ltree)                                ; lokale Var. 'ltree'
```

```
    (loop                                ; Schleife ueber Liste
      (setq ltree (pop stree))                  ; erster Ast nehmen
      (if (null ltree)                   ; Wenn letzter Ast gewesen
      (return)                           ;  dann Ende Rekursion
      (progn                              ;  sonst ...
        (setq *lncnt2 (1+ *lncnt2))             ; Ausgabezeilen zaehlen
          (format *outstream                    ; Ausgabe Tracetiefe
                  "~A~2D~A"
                  "(" level ")")
        (princ (make-string level               ; Einruecken
                  :initial-element #\Space)
               *outstream)
        (write-line (car ltree) *outstream)     ; Ausgabe Tracetag
        (if (not (null (cadr ltree)))           ; Wenn es kein Blatt ist,
          (SHOWTREE (cadr ltree) (1+ level))    ;  dann Rekursion in den Ast
            (setq *pathcnt (1+ *pathcnt))       ;  sonst Pfadzahl erhoehen
) ) ) ) ) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Abschlussarbeiten
;-------------------------------------------------------------------------
; Parameter                 : -
; Resultat                  : -
; Veraenderte glob. Variabeln : *traceline, *outstream, *instream
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun FINISH ()
  (loop                                 ; kopiere restliche Zeilen
    (cond
      ((equal *traceline "*EOF") (return))      ; Dateiende -> fertig
      (T (write-line *traceline *outstream))    ; sonst Zeile kopieren
    )
    (setq *traceline (read-line *instream nil "*EOF")) ; naechste Z. lesen
  )
  (format (make-broadcast-stream                ; Statistiken:
            *outstream                          ;  - Anzeige am Bildschirm
            *standard-output*)                  ;  - Ausgabe in Datei
          "~A ~A ~A~%~A ~A~%~A ~2D% (~D -> ~D)~%~A ~D~%"
          "Tree reduction ended :"
            (DATE)
            (CLOCK)
          "Reduction time       :"
            (CLOCK *time-elapsed)
          "Reduction factor     :"
            (- 100 (truncate (/ (* *lncnt2 100) *lncnt)))
            *lncnt *lncnt2
          "Count of paths       :"
            *pathcnt)
  (close *outstream)                            ;  Dateien schliessen
  (close *instream)
  (setq *result nil)
)
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; M A I N
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
;--- Vorbereitungen ------------------------------------------------------
;
  (format t "~A~%~A~%~A~%~%~A~%~%"
          "******************"
          "*    REDUC - 1   *"
          "******************"
          "(TTCN tree reduction algorithm 1)")
  (setq *lncnt 0
        *lncnt2 0
        *pathcnt 0
        *read-error nil)
  (when (null (SELECTFILES))                    ; I/O-Dateien bestimmen
    (write-line "User abort.")
    (return-from reduc)
```

```
  )
  (when (null (OPENFILES))                       ; Oeffnen der Dateien
    (close *instream) (close *outstream)
    (write-line "Error: could not open files")
    (return-from reduc)
  )
;
;--- Beginn Rekursion, Aufbau Trace-Baum ----------------------------------
;
  (setq *time-elapsed (get-universal-time))      ; Merke Startzeit
  (GETTRACELINE)
  (setq *result (BUILDTREE '(nil) 1))
  (setq *time-elapsed (- (get-universal-time)    ; Berechne verstrichene
                *time-elapsed))              ;  Zeit
;
;--- Abschlussarbeiten-----------------------------------------------------
;
  (write-line "Writing reduced tree ...")
  (SHOWTREE *result 1)                          ; Ausgabe Trace-Baum
  (write-line "Ok.")
  (FINISH)
)
;
;%%% E O F %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

D.2 Algorithmus REDUC-2

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; LISP-Algorithmus zur Reduktion eines Tracebaumes
; - Eingabe : Tracebaum              (Textfile)
; - Ausgabe : Korrigierter Tracebaum  (Textfile)
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(in-package "USER")
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Globale Variablen
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defvar *traceline "Start")        ; vollst. gelesene Zeile      (sequence)
(defvar *tracelevel 0)             ; Tiefe des Traces            (numeric)
(defvar *tracetag "None")          ; Trace-Tag                   (sequence)
(defvar *result nil)               ; Resultierender Trace-Baum   (list)
(defvar *lncnt 0)                  ; Anzahl eingelesener Zeilen  (numeric)
(defvar *lncnt2 0)                 ; Anzahl ausgegebener Zeilen  (numeric)
(defvar *pathcnt 0)                ; Anzahl ausgegebener Pfade   (numeric)
(defvar *instream nil)             ; Eingabe-Handle              (stream)
(defvar *outstream nil)            ; Ausgabe-Handle              (stream)
(defvar *read-error nil)           ; Lese-Fehler-Flag            (boolean)
(defvar *infilename "")            ; Name Eingabe-Datei          (sequence)
(defvar *outfilename "")           ; Name Ausgabe-Datei          (sequence)
(defvar *time-elapsed 0)           ; Zeitaufwand Reduktion       (numeric)
;
(defun REDUC2 ()
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Verschiedene kleine Hilfsfunktionen
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun CLOCK (&optional universal-time)
  "Return a string telling current time."
  (multiple-value-bind (sec min hrs)
    (decode-universal-time (if (null universal-time)
       (get-universal-time) universal-time))
    (format nil "~2,'0D:~2,'0D:~2,'0D"
          (if universal-time (1- hrs) hrs) min sec)
) )

(defun DATE ()
```

```
  "Return a string telling current date."
  (multiple-value-bind (sec min hrs date month year)
    (decode-universal-time (get-universal-time))
    (declare (ignore sec min hrs))
    (format nil "~2,'0D-~2,'0D-~2,'0D"
          date month (mod year 100)
) ) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Bestimmen der Input- und Outputfiles
;-------------------------------------------------------------------------------
; Parameter                  : keine
; Resultat                   : T = Ok / NIL = Fehler
; Veraenderte glob. Variabeln : *infilename, *outfilename
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun SELECTFILES ()
  (loop                                ; Schleife fuer Eingabefehler
    (terpri) (princ "Input file  : ")
    (setq *infilename (read-line))
    (when (equal *infilename "")               ; Wenn keine Eingabe
       (return-from selectfiles nil))          ;  dann Abbruch
    (when (probe-file *infilename)             ; Wenn Datei existiert,
       (return))                               ;  dann beenden der Schleife
    (terpri)                                   ;  sonst Fehlermeldung
    (princ "Error: input file does not exist !")
  )
  (loop                                ; Schleife fuer Eingabefehler
    (terpri) (princ "Output file : ")
    (setq *outfilename (read-line))
    (cond
      ((equal *outfilename "")                 ; Abbrechen ?
       (return-from selectfiles nil))
      ((probe-file *outfilename)               ; Datei existiert ?
       (princ "Replacing existing output file (y/N) : ")
       (when (equalp (read-line) "y")          ; Datei ueberschreiben ?
         (return T)))
      (T (return T))                           ; Datei neu, fertig
) ) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Oeffnen der Eingabe- & Ausgabedatei
;-------------------------------------------------------------------------------
; Parameter                  : keine
; Resultat                   : T = Ok / NIL = Fehler
; Veraenderte glob. Variabeln : *instream, *traceline
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun OPENFILES ()
  (setq *outstream                         ; Setzen des Ausgabestroms
       (open *outfilename :direction :output    ; auf die Ausgabedatei
             :if-exists :supersede))
  (format t "~%~A~%" "Reading input file ...")
  (setq *instream (open *infilename :direction :input)) ; Oeffnen der Datei
  (setq *traceline (read-line *instream nil "End of File"))
  (when (string-equal *traceline "REDUCED" :end1 7)
    (write-line "Input file is already a reduced version !")
    (princ "Continue anyway [y/N] : ")
    (when (not (equalp (read-line) "j"))
      (write-line "Program aborted.")
      (return-from reduc2 nil)
  )  )
  (format *outstream "~A~:@(~A~)~%"            ; Ausgabedatei markieren,
       "REDUC-2: This is a REDUCED version of file"
       *infilename)
  (write-line *traceline *outstream)
  (loop                                ; Einlese-Schleife
    (setq *traceline (read-line *instream       ; Zeile lesen solange
                     nil "End of File"))      ; Dateiende nicht erreicht
    (write-line *traceline *outstream)
    (cond
```

```
    ((< (length *traceline) 5))
    ((string-equal *traceline "start" :end1 5) ; Wenn erster Trace gefunden
     (return T))                          ;  dann beende Schleife
    ((string-equal *traceline "end" :end1 3)   ; Wenn Dateiende erreicht
     (terpri)
     (princ "Eror: No trace lines found !")    ; -> Fehlermeldung
     (return-from openfiles nil)              ;  und Abbruch
) ) ) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Einlesen einer Textzeile und Parsen nach Trace-Level und Trace-Tag
;-------------------------------------------------------------------------
; Parameter                  : keine
; Resultat                   : ohne Bedeutung
; Veraenderte glob. Variabeln : *traceline, *tracelevel, *tracetag
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun GETTRACELINE ()
  (setq *traceline (read-line *instream nil "*EOF"))
  (cond
   ((< (length *traceline) 4)
     (write-line "Reading error: ignoring invalid line"))
   ((string-equal *traceline "End" :end1 3)
     (format t "~A~A~%" #\Return
            "End of input file found")
     (setq *read-error T))
   ((string-equal *traceline "*EOF" :end1 4)
     (format t "~A~A~D~%" #\Return "line : " *lncnt)
     (write-line " Reading error: unexpected end of file !")
     (setq *read-error T))
   (T(setq *tracetag (string-trim " " (subseq *traceline 5)))
     (setq *tracelevel (car (read-from-string *traceline t nil)))
     (setq *lncnt (1+ *lncnt))
     (when (= (mod *lncnt 250) 0)
     (format t "~A~D" #\Return *lncnt)
  ) ) )
  (when *read-error
   (setq *tracetag "")
   (setq *tracelevel 0)
) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Reduktionsalgorithmus:
;   Entfernen aller doppelten, identischen Unterbaeume
;-------------------------------------------------------------------------
; Parameter                  : Baum
; Resultat                   : reduzierter Baum
; Veraenderte glob. Variabeln : -
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun CUT (tree)
  (let (branch hvar result)
   (setq result (list (pop tree)))             ; ersten Ast uebernehmen
   (loop                                  ; Schleife ueber alle Aeste
     (setq branch (pop tree))                  ; nimm naechsten Ast
     (if (null branch) (return result))        ; kein Ast mehr ?
     (if (not (and
          (setq hvar
          (assoc (car branch)                 ; Ast schon vorhanden ?
               result
             :test 'equal)
          )
        (tree-equal branch                    ; Und auch identisch ?
              hvar
              :test 'equal)
      ))
     (setq result (append result (list branch))) ; anfuegen
   ) )
   result
) )
;
```

```
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Rekursive Funktion:
; Einlesen der Trace-Zeilen mit gleichzeitigem Aufbau des Trace-Baumes
;-------------------------------------------------------------------------------
; Parameter                 : bisheriger Unterbaum, aktueller Tracelevel
; Resultat                  : neuer/korrigierter Unterbaum
; Veraenderte glob. Variabeln : [*traceline, *tracelevel, *tracetag]
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun BUILDTREE (subtree actlevel)
  (let ((localtag *tracetag) (treearg nil))     ; lokale Var. definieren
    (loop                                       ; Rekursions-Schleife
      (when (> actlevel *tracelevel)            ; Ende der Rekursion ?
      (setq subtree (CUT subtree))              ; wenn ja, dann reduziere
      (return))                                 ;  den erstellten U'Baum
      (if (= actlevel *tracelevel)              ; Verarb. auf dieser Ebene ?
      (progn                                    ; Wenn ja, dann ...
        (if (equal subtree '(nil))
          (setq subtree nil))
        (setq subtree (acons *tracetag '(nil)   ; einfuegen eines Astes
                     subtree))
        (setq localtag *tracetag)               ; Merke akt. Ast
        (GETTRACELINE)
      )
      (progn                                    ; Wenn nein, dann ...
        (setq treearg (cadr (assoc localtag     ; Unterbaum des aktuellen
                          subtree               ;  Astes bestimmen
                          :test #'equal)))
        (setq treearg (BUILDTREE treearg        ; Rekursions-Aufruf
                       (1+ actlevel)))
        (setq subtree (subst (cons localtag (list treearg))
                       (assoc localtag          ; Resultierender Unterbaum
                         subtree                ;  in 'subtree' ersetzen
                         :test #'equal)
                       subtree))
    ) ) )
    subtree                                     ; Unterbaum zurueckgeben
) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Ausgabe des resultierenden Trace-Baumes
;-------------------------------------------------------------------------------
; Parameter                 : Trace-Baum, Start-Trace-Level
; Resultat                  : ohne Bedeutung
; Veraenderte glob. Variabeln :  -
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun SHOWTREE (stree level)
  (let (ltree                                   ; lokale Var. 'ltree'
    (loop                                       ; Schleife ueber Liste
      (setq ltree (pop stree))                  ; erster Ast nehmen
      (if (null ltree)                          ; Wenn letzter Ast gewesen
      (return)                                  ;  dann Ende Rekursion
      (progn                                    ;  sonst ...
        (setq *lncnt2 (1+ *lncnt2))             ; Ausgabezeilen zaehlen
        (format *outstream                      ; Ausgabe Tracetiefe
              "~A~2D~A"
              "(" level ")")
        (princ (make-string level               ; Einruecken
                :initial-element #\Space)
               *outstream)
        (write-line (car ltree) *outstream)     ; Ausgabe Tracetag
        (if (not (null (cadr ltree)))           ; Wenn es kein Blatt ist,
          (SHOWTREE (cadr ltree) (1+ level))    ;  dann Rekursion in den Ast
          (setq *pathcnt (1+ *pathcnt))         ;  sonst Pfadzahl erhoehen
) ) ) ) ) )
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; Abschlussarbeiten
;-------------------------------------------------------------------------------
; Parameter                 : -
```

```
; Resultat                    : -
; Veraenderte glob. Variabeln : *traceline, *outstream, *instream
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
(defun FINISH ()
  (loop                                   ; kopiere restliche Zeilen
    (cond
      ((equal *traceline "*EOF") (return))       ; Dateiende -> fertig
      (T (write-line *traceline *outstream))     ; sonst Zeile kopieren
    )
    (setq *traceline (read-line *instream nil "*EOF")) ; naechste Z. lesen
  )
  (format (make-broadcast-stream               ; Statistiken:
           *outstream                         ;  - Anzeige am Bildschirm
           *standard-output*)                 ;  - Ausgabe in Datei
         "~A ~A ~A~%~A ~A~%~A ~2D% (~D -> ~D)~%~A ~D~%"
         "Tree reduction ended :"
          (DATE)
          (CLOCK)
         "Reduction time       :"
          (CLOCK *time-elapsed)
         "Reduction factor     :"
          (- 100 (truncate (/ (* *lncnt2 100) *lncnt)))
          *lncnt *lncnt2
         "Count of paths       :"
          *pathcnt)
  (close *outstream)                      ;  Dateien schliessen
  (close *instream)
  (setq *result nil)
)
;
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
; M A I N
;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
;
;--- Vorbereitungen -------------------------------------------------------
;
  (format t "~A~%~A~%~A~%~%~A~%~%"
         "******************"
         "*    REDUC - 2   *"
         "******************"
         "(TTCN tree reduction algorithm 2)")
  (setq *lncnt 0
       *lncnt2 0
       *pathcnt 0
       *read-error nil)
  (when (null (SELECTFILES))                  ; I/O-Dateien bestimmen
    (write-line "User abort.")
    (return-from reduc2)
  )
  (when (null (OPENFILES))                    ; Oeffnen der Dateien
    (close *instream) (close *outstream)
    (write-line "Error: could not open files")
    (return-from reduc2)
  )
;--- Beginn Rekursion, Aufbau Trace-Baum ----------------------------------
;
  (setq *time-elapsed (get-universal-time))      ; Merke Startzeit
  (GETTRACELINE)
  (setq *result (BUILDTREE '(nil) 1))
  (setq *time-elapsed (- (get-universal-time)    ; Berechne verstrichene
                 *time-elapsed))               ;  Zeit
;
;--- Abschlussarbeiten-----------------------------------------------------
;
  (write-line "Writing reduced tree ...")
  (SHOWTREE *result 1)                        ; Ausgabe Trace-Baum
  (write-line "Ok.")
  (FINISH))

;%%% E O F %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Anhang E Beispiele für reduzierte TESDL Testdaten

Wegen des Umfangs der generierten und reduzierten Testdaten, können hier nicht alle Daten ausgedruckt werden. In diesem Anhang sind nur die reduzierten Versionen der in Anhang C ausgedruckten TESDL Dateien zu finden.

E.1 REDUC-1: Verbindungsaufbau für die Testkonfiguration 3

REDUC-1: This is a REDUCED version of file AUFBAU.10
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 10
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-12 19:42:28

```
( 1) lt!MDATreq[DR,,,]
( 2) ut?IDISind
( 3) lt!MDATreq[DR,,,]
( 4) ut?IDISind
( 5) lt!MDATreq[DR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!ICONreq
( 8) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
( 5) ut!ICONreq
( 6) lt?MDATind[CR,,,]
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) lt!MDATreq[CC,,,]
( 8) ut?ICONconf
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 7) lt?MDATind[CR,,,]
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) lt!MDATreq[CC,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
(10) ut!IDATreq[,0,0]
( 8) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
( 5) lt!MDATreq[CC,,,]
( 6) ut?ICONconf
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!IDATreq[,0,0]
( 8) lt?MDATind[DT,TRUE,0,0]
( 9) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 9) lt!MDATreq[AK,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) ut!IDATreq[,0,0]
( 9) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[AK,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 5) lt?MDATind[CR,,,]
( 6) lt!MDATreq[DR,,,]
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut!ICONreq
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
( 6) lt!MDATreq[CC,,,]
( 7) ut?ICONconf
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut!IDATreq[,0,0]
( 9) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[AK,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 6) lt?MDATind[CR,,,]
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) lt!MDATreq[CC,,,]
( 8) ut?ICONconf
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 7) lt?MDATind[CR,,,]
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) lt!MDATreq[CC,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
(10) ut!IDATreq[,0,0]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 1) ut!ICONreq
( 2) lt?MDATind[CR,,,]
( 3) lt!MDATreq[DR,,,]
( 4) ut?IDISind
( 5) lt!MDATreq[DR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!ICONreq
( 8) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
( 5) ut!ICONreq
( 6) lt?MDATind[CR,,,]
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) lt!MDATreq[CC,,,]
( 8) ut?ICONconf
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 7) lt?MDATind[CR,,,]
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) lt!MDATreq[CC,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
(10) ut!IDATreq[,0,0]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) lt!MDATreq[CC,,,]
( 8) ut?ICONconf
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 7) lt?MDATind[CR,,,]
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
```

```
( 8)      lt!MDATreq[CC,,,]
( 9)       ut?ICONconf
(10)         lt!MDATreq[DR,,,]
(10)         ut!IDATreq[,0,0]
( 8)      lt?MDATind[CR,,,]
( 9)       lt!MDATreq[DR,,,]
(10)         ut?IDISind
( 9)       lt!MDATreq[CC,,,]
(10)         ut?ICONconf
( 9)       lt?MDATind[CR,,,]
(10)         lt!MDATreq[DR,,,]
(10)         lt!MDATreq[CC,,,]
(10)         ut?IDISind
( 3)   lt!MDATreq[CC,,,]
( 4)    ut?ICONconf
( 5)     lt!MDATreq[DR,,,]
( 6)      ut?IDISind
( 7)       lt!MDATreq[DR,,,]
( 8)        ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)         ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 7)       ut!ICONreq
( 8)        lt?MDATind[CR,,,]
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)         lt!MDATreq[CC,,,]
(10)           ut?ICONconf
( 9)         lt?MDATind[CR,,,]
(10)           lt!MDATreq[DR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt?MDATind[CR,,,]
( 5)     ut!IDATreq[,0,0]
( 6)      lt?MDATind[DT,TRUE,0,0]
( 7)       lt!MDATreq[DR,TRUE,0,0]
( 8)        lt!MDATreq[DR,TRUE,0,0]
( 9)         lt!MDATreq[DR,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
( 9)         lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
( 8)        lt?MDATind[DT,TRUE,0,0]
( 9)         lt!MDATreq[DR,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
( 9)         lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
( 7)       lt!MDATreq[AK,TRUE,0,0]
( 8)        lt!MDATreq[DR,TRUE,0,0]
( 9)         ut?IDISind
(10)           lt!MDATreq[DR,TRUE,0,0]
(10)           ut!ICONreq
( 8)        ut!IDATreq[,0,0]
( 9)         lt?MDATind[DT,FALSE,0,0]
(10)           lt!MDATreq[DR,FALSE,0,0]
(10)           lt!MDATreq[AK,FALSE,0,0]
(10)           lt?MDATind[DT,FALSE,0,0]
( 7)       lt?MDATind[DT,TRUE,0,0]

( 8)      lt!MDATreq[DR,TRUE,0,0]
( 9)       lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 9)       lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 8)      lt!MDATreq[AK,TRUE,0,0]
( 9)       lt!MDATreq[DR,TRUE,0,0]
(10)         ut?IDISind
( 9)       ut!IDATreq[,0,0]
(10)         lt?MDATind[DT,FALSE,0,0]
( 8)      lt?MDATind[DT,TRUE,0,0]
( 9)       lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 9)       lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 8)      lt!MDATreq[AK,TRUE,0,0]
( 9)       lt!MDATreq[DR,TRUE,0,0]
(10)         ut?IDISind
( 9)       ut!IDATreq[,0,0]
(10)         lt?MDATind[DT,FALSE,0,0]
( 8)      lt?MDATind[DT,TRUE,0,0]
( 9)       lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 9)       lt!MDATreq[AK,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt!IDATreq[,0,0]
( 9)       lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[AK,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 4)    lt?MDATind[CR,,,]
( 5)     lt!MDATreq[DR,,,]
( 6)      ut?IDISind
( 7)       lt!MDATreq[DR,,,]
( 8)        ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)         lt!MDATreq[CC,,,]
(10)           ut?ICONconf
( 9)         lt?MDATind[CR,,,]
(10)           lt!MDATreq[DR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt?MDATind[CR,,,]
( 6)       ut!ICONreq
( 7)        lt?MDATind[CR,,,]
( 8)         lt!MDATreq[DR,,,]
( 9)          ut?IDISind
(10)            lt!MDATreq[DR,,,]
(10)            ut!ICONreq
( 8)         lt!MDATreq[CC,,,]
( 9)          ut?ICONconf
(10)            lt!MDATreq[DR,,,]
(10)            ut!IDATreq[,0,0]
( 8)         lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)            ut?IDISind
( 9)          lt!MDATreq[CC,,,]
(10)            ut?ICONconf
( 9)          lt?MDATind[CR,,,]
(10)            lt!MDATreq[DR,,,]
(10)            lt!MDATreq[CC,,,]
(10)            lt?MDATind[CR,,,]
( 4)     lt!MDATreq[CC,,,]
( 5)      ut?ICONconf
( 6)       lt!MDATreq[DR,,,]
( 7)        ut?IDISind
( 8)         lt!MDATreq[DR,,,]

( 9)        ut?IDISind
(10)          lt!MDATreq[DR,,,]
(10)          ut!ICONreq
( 8)        ut!ICONreq
( 9)         lt?MDATind[CR,,,]
(10)           lt!MDATreq[DR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt?MDATind[CR,,,]
( 6)      ut!IDATreq[,0,0]
( 7)       lt?MDATind[DT,TRUE,0,0]
( 8)        lt!MDATreq[DR,TRUE,0,0]
( 9)         lt!MDATreq[DR,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
( 9)         lt?MDATind[DT,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
( 8)        lt!MDATreq[AK,TRUE,0,0]
( 9)         lt!MDATreq[DR,TRUE,0,0]
(10)           ut?IDISind
( 9)         ut!IDATreq[,0,0]
(10)           lt?MDATind[DT,FALSE,0,0]
( 8)        lt?MDATind[DT,TRUE,0,0]
( 9)         lt!MDATreq[DR,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
( 9)         lt!MDATreq[AK,TRUE,0,0]
(10)           lt!MDATreq[DR,TRUE,0,0]
(10)           ut!IDATreq[,0,0]
(10)           lt?MDATind[DT,TRUE,0,0]
( 4)     lt?MDATind[CR,,,]
( 5)      lt!MDATreq[DR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)            ut?IDISind
( 9)          ut!ICONreq
(10)            lt?MDATind[CR,,,]
( 7)        ut!IDATreq[,0,0]

( 8)      lt?MDATind[DT,TRUE,0,0]
( 9)       lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 9)       lt!MDATreq[AK,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         ut!IDATreq[,0,0]
( 9)       lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 6)      ut?IDISind
( 7)       lt!MDATreq[DR,,,]
( 8)        ut?IDISind
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)         ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 7)       lt!MDATreq[CC,,,]
( 8)        lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)           lt!MDATreq[DR,,,]
(10)           ut!ICONreq
( 8)        ut!ICONreq
( 9)         lt?MDATind[CR,,,]
(10)           lt!MDATreq[DR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt?MDATind[CR,,,]
( 7)       ut!ICONreq
( 8)        lt?MDATind[CR,,,]
( 9)         lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)         lt!MDATreq[CC,,,]
(10)           ut?ICONconf
( 9)         lt?MDATind[CR,,,]
(10)           lt!MDATreq[DR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt?MDATind[CR,,,]
```

End of generation: 92-05-12 21:35:16
End of Testcases
Tree reduction ended : 02-04-93 11:48:55
Reduction time : 00:00:07
Reduction factor : 74% (1639 -> 432)
Count of paths : 181

E.2 REDUC-2: Verbindungsaufbau für die Testkonfiguration 3

REDUC-2: This is a REDUCED version of file AUFBAU.10
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 10
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-12 19:42:28

```
( 1) lt!MDATreq[DR,,,]
( 2) ut?IDISind
( 3) lt!MDATreq[DR,,,]
( 4) ut?IDISind
( 5) lt!MDATreq[DR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!ICONreq
( 8) lt?MDATind[CR,,,]
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 4) ut?IDISind
( 5) lt!MDATreq[DR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!ICONreq
( 8) lt?MDATind[CR,,,]
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
```

```
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 5) ut!ICONreq
( 6) lt?MDATind[CR,,,]
( 7) lt?MDATind[CR,,,]
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) lt!MDATreq[CC,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
(10) ut!IDATreq[,0,0]
( 8) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) lt!MDATreq[CC,,,]
( 8) ut?ICONconf
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?ICONconf
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 2) ut?IDISind
( 3) lt!MDATreq[DR,,,]
( 4) ut?IDISind
( 5) lt!MDATreq[DR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
```

```
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!ICONreq
( 8) lt?MDATind[CR,,,]
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 4) ut?IDISind
( 5) lt!MDATreq[DR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!ICONreq
( 8) lt?MDATind[CR,,,]
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 5) ut!ICONreq
( 6) lt?MDATind[CR,,,]
( 7) lt?MDATind[CR,,,]
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) lt!MDATreq[CC,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
```

```
(10) ut!IDATreq[,0,0]
( 8) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 3) ut!ICONreq
( 4) lt?MDATind[CR,,,]
( 5) lt?MDATind[CR,,,]
( 6) lt!MDATreq[DR,,,]
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut!ICONreq
( 9) lt?MDATind[CR,,,]
(10) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
( 6) lt!MDATreq[CC,,,]
( 7) ut?ICONconf
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
```

```
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut!IDATreq[,0,0]
( 9) lt?MDATind[DT,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[AK,TRUE,0,0]
( 6) lt?MDATind[CR,,,]
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) lt!MDATreq[CC,,,]
( 8) ut?ICONconf
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?ICONconf
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 7) lt?MDATind[CR,,,]
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) lt!MDATreq[CC,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
(10) ut!IDATreq[,0,0]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 5) lt!MDATreq[DR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!ICONreq
( 8) lt?MDATind[CR,,,]
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 5) lt!MDATreq[CC,,,]
( 6) ut?ICONconf
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 6) ut?ICONconf
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!IDATreq[,0,0]
( 8) lt?MDATind[DT,TRUE,0,0]
( 9) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[AK,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 9) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 9) lt!MDATreq[AK,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) ut!IDATreq[,0,0]
( 1) ut!ICONreq
( 2) lt?MDATind[CR,,,]
( 3) lt?MDATind[CR,,,]
( 4) lt!MDATreq[DR,,,]
( 5) ut?IDISind
( 6) lt!MDATreq[DR,,,]
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut!ICONreq
( 9) lt?MDATind[CR,,,]
(10) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
( 5) ut?IDISind
( 6) lt!MDATreq[DR,,,]
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut!ICONreq
( 9) lt?MDATind[CR,,,]
(10) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
( 6) ut!ICONreq
( 7) lt?MDATind[CR,,,]
( 8) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) lt!MDATreq[CC,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
( 9) ut?ICONconf
(10) lt!MDATreq[DR,,,]
(10) ut!IDATreq[,0,0]
( 4) lt!MDATreq[CC,,,]
( 5) ut?ICONconf
( 6) lt!MDATreq[DR,,,]
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut!ICONreq
( 9) lt?MDATind[CR,,,]
(10) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
( 5) ut?ICONconf
( 6) lt!MDATreq[DR,,,]
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut!ICONreq
( 9) lt?MDATind[CR,,,]
(10) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
( 6) ut!IDATreq[,0,0]
( 7) lt?MDATind[DT,TRUE,0,0]
( 8) lt?MDATind[DT,TRUE,0,0]
( 9) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 9) lt!MDATreq[AK,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) ut!IDATreq[,0,0]
( 9) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[AK,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 8) lt!MDATreq[DR,TRUE,0,0]
( 9) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 9) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 8) lt!MDATreq[AK,TRUE,0,0]
( 9) lt!MDATreq[DR,TRUE,0,0]
(10) ut?IDISind
( 9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,FALSE,0,0]
( 4) lt?MDATind[CR,,,]
( 5) lt!MDATreq[DR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 6) ut?IDISind
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!ICONreq
( 8) lt?MDATind[CR,,,]
( 9) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
(10) lt?MDATind[CR,,,]
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) lt!MDATreq[CC,,,]
(10) ut?ICONconf
( 5) lt!MDATreq[CC,,,]
( 6) ut?ICONconf
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 6) ut?ICONconf
( 7) lt!MDATreq[DR,,,]
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 8) ut?IDISind
( 9) lt!MDATreq[DR,,,]
(10) ut?IDISind
( 9) ut!ICONreq
(10) lt?MDATind[CR,,,]
( 7) ut!IDATreq[,0,0]
( 8) lt?MDATind[DT,TRUE,0,0]
( 9) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[AK,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 9) lt!MDATreq[DR,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
( 9) lt!MDATreq[AK,TRUE,0,0]
(10) lt!MDATreq[DR,TRUE,0,0]
(10) ut!IDATreq[,0,0]
( 5) lt?MDATind[CR,,,]
( 6) lt!MDATreq[DR,,,]
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 7) ut?IDISind
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut!ICONreq
( 9) lt?MDATind[CR,,,]
(10) lt?MDATind[CR,,,]
(10) lt!MDATreq[DR,,,]
(10) lt!MDATreq[CC,,,]
( 6) lt!MDATreq[CC,,,]
( 7) ut?ICONconf
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 7) ut?ICONconf
( 8) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
( 9) ut?IDISind
(10) lt!MDATreq[DR,,,]
(10) ut!ICONreq
( 8) ut!IDATreq[,0,0]
( 9) lt?MDATind[DT,TRUE,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
```

```
(10)       lt!MDATreq[DR,TRUE,0,0]
(10)       lt!MDATreq[AK,TRUE,0,0]
( 6)     ut?IDISind
( 7)      lt!MDATreq[DR,,,]
( 8)       ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
( 8)       ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
( 9)        ut!ICONreq
(10)         lt?MDATind[CR,,,]
( 6)     ut?IDISind
( 7)      lt!MDATreq[CC,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)         lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)         lt!MDATreq[DR,,,]
(10)         ut!ICONreq
( 8)       ut!ICONreq
( 9)        lt?MDATind[CR,,,]
(10)         lt?MDATind[CR,,,]
(10)         lt!MDATreq[DR,,,]
(10)         lt!MDATreq[CC,,,]
( 6)     ut?IDISind
( 7)      lt!MDATreq[DR,,,]
( 8)       ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
( 8)       ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
( 9)        ut!ICONreq
(10)         lt?MDATind[CR,,,]
( 7)      lt!MDATreq[CC,,,]
( 8)       lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)         lt!MDATreq[DR,,,]
( 9)        ut?IDISind
(10)         lt!MDATreq[DR,,,]
(10)         ut!ICONreq
( 8)       ut!ICONreq
( 9)        lt?MDATind[CR,,,]
(10)         lt?MDATind[CR,,,]
(10)         lt!MDATreq[DR,,,]
(10)         lt!MDATreq[CC,,,]
( 7)       ut!ICONreq
( 8)        lt?MDATind[CR,,,]
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
( 9)        lt!MDATreq[CC,,,]
(10)         ut?ICONconf
( 9)        lt?MDATind[CR,,,]
(10)         lt!MDATreq[DR,,,]
(10)         lt!MDATreq[CC,,,]
(10)         lt?MDATind[CR,,,]
( 3)   lt!MDATreq[DR,,,]
( 4)     ut?IDISind
( 5)      lt!MDATreq[DR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 7)        lt!MDATreq[CC,,,]
( 8)         lt!MDATreq[DR,,,]
( 9)          ut?IDISind
(10)           lt!MDATreq[DR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt?MDATind[CR,,,]
( 3)   lt!MDATreq[DR,,,]
( 4)     ut?IDISind
( 5)      lt!MDATreq[DR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 7)        ut!ICONreq
( 8)         lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          lt!MDATreq[CC,,,]
(10)           ut?ICONconf
( 5)     ut!ICONreq
( 6)      lt?MDATind[CR,,,]
( 7)       lt?MDATind[CR,,,]
( 8)        lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          lt!MDATreq[DR,,,]
( 9)         ut?IDISind
(10)          lt!MDATreq[DR,,,]
(10)          ut!ICONreq
( 8)        lt!MDATreq[CC,,,]
( 9)         ut?ICONconf
(10)          lt!MDATreq[DR,,,]
( 9)         ut?ICONconf
(10)          lt!MDATreq[DR,,,]
(10)          ut!IDATreq[,0,0]
( 8)        lt?MDATind[CR,,,]
( 9)         lt!MDATreq[DR,,,]
(10)          ut?IDISind
( 9)         lt!MDATreq[CC,,,]
(10)          ut?ICONconf
( 4)     ut?IDISind
( 5)      lt!MDATreq[DR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 7)        ut!ICONreq
( 8)         lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          lt!MDATreq[CC,,,]
(10)           ut?ICONconf
( 4)     ut?ICONconf
( 5)      lt!MDATreq[DR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 7)        ut!ICONreq
( 8)         lt?MDATind[CR,,,]
( 9)          lt?MDATind[CR,,,]
(10)           lt!MDATreq[DR,,,]
(10)           lt!MDATreq[CC,,,]
(10)           lt?MDATind[CR,,,]
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          lt!MDATreq[CC,,,]
(10)           ut?ICONconf
( 4)     ut?ICONconf
( 5)      lt!MDATreq[DR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)       ut?IDISind
( 8)       ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
( 9)        ut!ICONreq
(10)         lt?MDATind[CR,,,]
( 7)      ut!ICONreq
( 8)       lt?MDATind[CR,,,]
( 9)        lt?MDATind[CR,,,]
(10)         lt!MDATreq[DR,,,]
(10)         lt!MDATreq[CC,,,]
(10)         lt?MDATind[CR,,,]
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
( 9)        lt!MDATreq[CC,,,]
(10)         ut?ICONconf
( 4)     ut?ICONconf
( 5)      lt!MDATreq[DR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)           ut?IDISind
( 9)          ut!ICONreq
(10)           lt?MDATind[CR,,,]
( 6)       ut?IDISind
( 7)        lt!MDATreq[DR,,,]
( 8)         ut?IDISind
( 9)          lt!MDATreq[DR,,,]
(10)       ut?IDISind
( 8)       ut?IDISind
( 9)        lt!MDATreq[DR,,,]
(10)         ut?IDISind
( 9)        ut!ICONreq
(10)         lt?MDATind[CR,,,]
( 7)      ut!ICONreq
( 8)       lt?MDATind[CR,,,]
( 9)        lt?MDATind[CR,,,]
(10)         lt!MDATreq[DR,,,]
(10)         lt!MDATreq[CC,,,]
(10)         lt?MDATind[CR,,,]
( 8)       lt?MDATind[DT,TRUE,0,0]
( 9)        lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 9)        lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 8)       lt!MDATreq[AK,TRUE,0,0]
( 9)        lt!MDATreq[DR,TRUE,0,0]
(10)         ut?IDISind
( 9)        ut!IDATreq[,0,0]
(10)         lt?MDATind[DT,FALSE,0,0]
( 8)       lt?MDATind[DT,TRUE,0,0]
( 9)        lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 9)        lt!MDATreq[AK,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         ut!IDATreq[,0,0]
( 9)        lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[AK,TRUE,0,0]
(10)         ut?IDISind
( 7)      lt!MDATreq[DR,TRUE,0,0]
( 8)       lt!MDATreq[DR,TRUE,0,0]
( 9)        lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 9)        lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 8)       lt?MDATind[DT,TRUE,0,0]
( 9)        lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt?MDATind[DT,TRUE,0,0]
( 9)        lt?MDATind[DT,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         lt!MDATreq[DR,TRUE,0,0]
( 7)      lt!MDATreq[AK,TRUE,0,0]
( 8)       lt!MDATreq[DR,TRUE,0,0]
( 9)        ut?IDISind
(10)         lt!MDATreq[DR,TRUE,0,0]
( 9)        ut?IDISind
(10)         lt!MDATreq[DR,TRUE,0,0]
(10)         ut!ICONreq
( 8)       ut!IDATreq[,0,0]
( 9)        lt?MDATind[DT,FALSE,0,0]
(10)         lt?MDATind[DT,FALSE,0,0]
(10)         lt!MDATreq[DR,FALSE,0,0]
(10)         lt!MDATreq[AK,FALSE,0,0]
```

End of generation: 92-05-12 21:35:16
End of Testcases
Tree reduction ended : 02-04-93 12:20:43
Reduction time : 00:00:05
Reduction factor : 46% (1639 -> 894)
Count of paths : 352

E.3 REDUC-1: Datenphase für die Testkonfiguration 1

```
REDUC-1: This is a REDUCED version of file DATEN.4
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE :          4
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : ENV, System-Process : Empfaenger
Signalroute : ISAP
REDUCED SIGNALS:
Process: Empfaenger, State: warten, Signal: IDISreq
Process: Empfaenger, State: verbunden, Signal: IDISreq
Process: Empfaenger, State: unterbrochen, Signal: IDISreq
Process: Empfaenger, State: verbunden1, Signal: IDISreq
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-12 17:46:27

( 1) ut!IDATreq[,0,0]
( 2)  lt?IDATind[0,0]
( 3)   ut!IDATreq[,0,0]
( 4)    lt?IDATind[0,0]
( 3)   ut?IDISind
( 4)    ut!ICONreq

End of generation: 92-05-12 17:49:04
End of Testcases
Tree reduction ended : 02-04-93 11:30:48
Reduction time       : 00:00:00
Reduction factor     : 83% (34 -> 6)
Count of paths       : 2
```

E.4 REDUC-2 Datenphase für die Testkonfiguration 1

```
REDUC-2: This is a REDUCED version of file DATEN.4
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE :          4
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : ENV, System-Process : Empfaenger
Signalroute : ISAP
REDUCED SIGNALS:
Process: Empfaenger, State: warten, Signal: IDISreq
Process: Empfaenger, State: verbunden, Signal: IDISreq
Process: Empfaenger, State: unterbrochen, Signal: IDISreq
Process: Empfaenger, State: verbunden1, Signal: IDISreq
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-12 17:46:27

( 1) ut!IDATreq[,0,0]
( 2)  lt?IDATind[0,0]
( 3)   ut!IDATreq[,0,0]
( 4)    lt?IDATind[0,0]
```

(3) ut?IDISind
(4) ut!ICONreq

End of generation: 92-05-12 17:49:04
End of Testcases
Tree reduction ended : 02-04-93 12:02:27
Reduction time : 00:00:00
Reduction factor : 83% (34 -> 6)
Count of paths : 2

E.5 REDUC-1: Datenphase für die Testkonfiguration 3

REDUC-1: This is a REDUCED version of file DATEN.11
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 11
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REDUCED SIGNALS:
Process: Empfaenger, State: warten, Signal: IDISreq
Process: Empfaenger, State: unterbrochen, Signal: IDISreq
Process: Empfaenger, State: verbunden, Signal: IDISreq
Process: Empfaenger, State: verbunden1, Signal: IDISreq
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-14 15:31:28

(1) ut!IDATreq[,0,0]
(2) lt?MDATind[DT,TRUE,0,0]
(3) lt!MDATreq[AK,TRUE,0,0]
(4) ut!IDATreq[,0,0]
(5) lt?MDATind[DT,FALSE,0,0]
(6) lt!MDATreq[AK,FALSE,0,0]
(7) ut!IDATreq[,0,0]
(8) lt?MDATind[DT,TRUE,0,0]
(9) lt!MDATreq[AK,TRUE,0,0]
(10) ut!IDATreq[,0,0]
(11) lt?MDATind[DT,FALSE,0,0]
(9) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[AK,TRUE,0,0]
(11) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
(11) lt!MDATreq[AK,TRUE,0,0]
(11) lt?MDATind[DT,TRUE,0,0]
(6) lt?MDATind[DT,FALSE,0,0]
(7) lt!MDATreq[AK,FALSE,0,0]
(8) ut!IDATreq[,0,0]
(9) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[AK,TRUE,0,0]
(11) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
(11) lt!MDATreq[AK,TRUE,0,0]
(11) lt?MDATind[DT,TRUE,0,0]
(7) lt?MDATind[DT,FALSE,0,0]
(8) lt!MDATreq[AK,FALSE,0,0]
(9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
(11) lt!MDATreq[AK,TRUE,0,0]
(11) lt?MDATind[DT,TRUE,0,0]
(8) lt?MDATind[DT,FALSE,0,0]
(9) lt!MDATreq[AK,FALSE,0,0]
(10) ut!IDATreq[,0,0]
(11) lt?MDATind[DT,TRUE,0,0]
(8) lt?MDATind[DT,FALSE,0,0]
(9) lt!MDATreq[AK,FALSE,0,0]
(10) ut!IDATreq[,0,0]
(11) ut!ICONreq
(10) ut!ICONreq
(11) lt?MDATind[CR,FALSE,0,0]
(3) lt?MDATind[DT,TRUE,0,0]
(4) lt!MDATreq[AK,TRUE,0,0]
(5) ut!IDATreq[,0,0]
(6) lt?MDATind[DT,FALSE,0,0]
(7) lt!MDATreq[AK,FALSE,0,0]
(8) ut!IDATreq[,0,0]
(9) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[AK,TRUE,0,0]
(11) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
(11) lt!MDATreq[AK,TRUE,0,0]
(11) lt?MDATind[DT,TRUE,0,0]
(7) lt?MDATind[DT,FALSE,0,0]
(8) lt!MDATreq[AK,FALSE,0,0]
(9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
(11) lt!MDATreq[AK,TRUE,0,0]
(11) lt?MDATind[DT,TRUE,0,0]
(8) lt?MDATind[DT,FALSE,0,0]
(9) lt!MDATreq[AK,FALSE,0,0]
(10) ut!IDATreq[,0,0]
(11) lt?MDATind[DT,TRUE,0,0]
(9) lt?MDATind[DT,FALSE,0,0]
(10) lt!MDATreq[AK,FALSE,0,0]
(11) ut!IDATreq[,0,0]
(10) ut?IDISind
(11) lt!MDATreq[AK,FALSE,0,0]
(11) ut!ICONreq
(4) lt?MDATind[DT,TRUE,0,0]
(5) lt!MDATreq[AK,TRUE,0,0]
(6) ut!IDATreq[,0,0]
(7) lt?MDATind[DT,FALSE,0,0]
(8) lt!MDATreq[AK,FALSE,0,0]
(9) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
(11) lt!MDATreq[AK,TRUE,0,0]
(11) lt?MDATind[DT,TRUE,0,0]
(8) lt?MDATind[DT,FALSE,0,0]
(9) lt!MDATreq[AK,FALSE,0,0]
(10) ut!IDATreq[,0,0]
(11) lt?MDATind[DT,TRUE,0,0]
(9) lt?MDATind[DT,FALSE,0,0]
(10) lt!MDATreq[AK,FALSE,0,0]
(11) ut!IDATreq[,0,0]
(10) lt?MDATind[DT,TRUE,0,0]
(11) lt!MDATreq[AK,FALSE,0,0]
(11) ut?IDISind
(5) lt?MDATind[DT,TRUE,0,0]
(6) lt!MDATreq[AK,TRUE,0,0]
(7) ut!IDATreq[,0,0]
(8) lt?MDATind[DT,FALSE,0,0]
(9) lt!MDATreq[AK,FALSE,0,0]
(10) ut!IDATreq[,0,0]
(11) lt?MDATind[DT,TRUE,0,0]
(9) lt?MDATind[DT,FALSE,0,0]
(10) lt!MDATreq[AK,FALSE,0,0]
(11) ut!IDATreq[,0,0]
(9) lt?MDATind[DT,TRUE,0,0]
(10) lt!MDATreq[AK,FALSE,0,0]
(11) lt?MDATind[DT,FALSE,0,0]
(6) ut?IDISind
(7) lt!MDATreq[AK,TRUE,0,0]
(8) ut!ICONreq
(9) lt?MDATind[CR,TRUE,0,0]
(10) lt!MDATreq[CC,TRUE,0,0]
(11) ut?ICONconf
(10) lt?MDATind[CR,TRUE,0,0]
(11) lt!MDATreq[CC,TRUE,0,0]
(11) lt?MDATind[CR,TRUE,0,0]
(7) ut!ICONreq
(8) lt?MDATind[CR,TRUE,0,0]
(9) lt!MDATreq[CC,TRUE,0,0]
(10) ut?ICONconf
(11) ut!IDATreq[,0,0]
(9) lt?MDATind[CR,TRUE,0,0]
(10) lt!MDATreq[CC,TRUE,0,0]
(11) ut?ICONconf
(10) lt?MDATind[CR,TRUE,0,0]
(11) lt!MDATreq[CC,TRUE,0,0]
(11) lt?MDATind[CR,TRUE,0,0]

End of generation: 92-05-14 16:18:09
End of Testcases
Tree reduction ended : 02-04-93 11:49:11
Reduction time : 00:00:01
Reduction factor : 68% (375 -> 121)
Count of paths : 38

E.6 REDUC-2: Datenphase für die Testkonfiguration 3

```
REDUC-2: This is a REDUCED version of file DATEN.11
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE :        11
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REDUCED SIGNALS:
Process: Empfaenger, State: warten, Signal: IDISreq
Process: Empfaenger, State: unterbrochen, Signal: IDISreq
Process: Empfaenger, State: verbunden, Signal: IDISreq
Process: Empfaenger, State: verbunden1, Signal: IDISreq
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-14 15:31:28
```

```
( 1) ut!IDATreq[,0,0]
( 2)  lt?MDATind[DT,TRUE,0,0]
( 3)   lt?MDATind[DT,TRUE,0,0]
( 4)    lt!MDATreq[AK,TRUE,0,0]
( 5)     ut!IDATreq[,0,0]
( 6)      lt?MDATind[DT,FALSE,0,0]
( 7)       lt?MDATind[DT,FALSE,0,0]
( 8)        lt!MDATreq[AK,FALSE,0,0]
( 9)         ut!IDATreq[,0,0]
(10)          lt?MDATind[DT,TRUE,0,0]
(11)           lt?MDATind[DT,TRUE,0,0]
(11)           lt!MDATreq[AK,TRUE,0,0]
( 8)        lt?MDATind[DT,FALSE,0,0]
( 9)         lt!MDATreq[AK,FALSE,0,0]
(10)          ut!IDATreq[,0,0]
(11)           lt?MDATind[DT,TRUE,0,0]
( 9)         lt?MDATind[DT,FALSE,0,0]
(10)          lt!MDATreq[AK,FALSE,0,0]
(11)           ut!IDATreq[,0,0]
(10)          ut?IDISind
(11)           lt!MDATreq[AK,FALSE,0,0]
(10)          ut?IDISind
(11)           lt!MDATreq[AK,FALSE,0,0]
(11)           ut!ICONreq
( 7)       lt!MDATreq[AK,FALSE,0,0]
( 8)        ut!IDATreq[,0,0]
( 9)         lt?MDATind[DT,TRUE,0,0]
(10)          lt?MDATind[DT,TRUE,0,0]
(11)           lt!MDATreq[AK,TRUE,0,0]
(11)           lt?MDATind[DT,TRUE,0,0]
(10)          lt!MDATreq[AK,TRUE,0,0]
(11)           ut!IDATreq[,0,0]
( 4)    lt?MDATind[DT,TRUE,0,0]
( 5)     lt!MDATreq[AK,TRUE,0,0]
( 6)      ut!IDATreq[,0,0]
( 7)       lt?MDATind[DT,FALSE,0,0]
( 8)        lt?MDATind[DT,FALSE,0,0]
( 9)         lt!MDATreq[AK,FALSE,0,0]
(10)          ut!IDATreq[,0,0]
(11)           lt?MDATind[DT,TRUE,0,0]
( 9)         lt?MDATind[DT,FALSE,0,0]
(10)          lt!MDATreq[AK,FALSE,0,0]
(11)           ut!IDATreq[,0,0]
(10)          lt?MDATind[DT,FALSE,0,0]
(11)           lt!MDATreq[AK,FALSE,0,0]
(11)           ut?IDISind
( 8)        lt!MDATreq[AK,FALSE,0,0]
( 9)         ut!IDATreq[,0,0]
(10)          lt?MDATind[DT,TRUE,0,0]
(11)           lt?MDATind[DT,TRUE,0,0]
(11)           lt!MDATreq[AK,TRUE,0,0]
( 5)     lt?MDATind[DT,TRUE,0,0]
( 6)      lt!MDATreq[AK,TRUE,0,0]
( 7)       ut!IDATreq[,0,0]
( 8)        lt?MDATind[DT,FALSE,0,0]
( 9)         lt?MDATind[DT,FALSE,0,0]
(10)          lt!MDATreq[AK,FALSE,0,0]
(11)           ut!IDATreq[,0,0]
(10)          lt?MDATind[DT,FALSE,0,0]
(11)           lt!MDATreq[AK,FALSE,0,0]
(11)           lt?MDATind[DT,FALSE,0,0]
( 9)         lt!MDATreq[AK,FALSE,0,0]
(10)          ut!IDATreq[,0,0]
(11)           lt?MDATind[DT,TRUE,0,0]
( 6)      ut?IDISind
( 7)       lt!MDATreq[AK,TRUE,0,0]
( 8)        ut!ICONreq
( 9)         lt?MDATind[CR,TRUE,0,0]
(10)          lt?MDATind[CR,TRUE,0,0]
(11)           lt!MDATreq[CC,TRUE,0,0]
(11)           lt?MDATind[CR,TRUE,0,0]
(10)          lt!MDATreq[CC,TRUE,0,0]
(11)           ut?ICONconf
( 6)      ut?IDISind
( 7)       lt!MDATreq[AK,TRUE,0,0]
( 8)        ut!ICONreq
( 9)         lt?MDATind[CR,TRUE,0,0]
(10)          lt?MDATind[CR,TRUE,0,0]
(11)           lt!MDATreq[CC,TRUE,0,0]
(11)           lt?MDATind[CR,TRUE,0,0]
(10)          lt!MDATreq[CC,TRUE,0,0]
(11)           ut?ICONconf
( 7)       ut!ICONreq
( 8)        lt?MDATind[CR,TRUE,0,0]
( 9)         lt?MDATind[CR,TRUE,0,0]
(10)          lt!MDATreq[CC,TRUE,0,0]
(11)           ut?ICONconf
(10)          lt?MDATind[CR,TRUE,0,0]
(11)           lt!MDATreq[CC,TRUE,0,0]
(11)           lt?MDATind[CR,TRUE,0,0]
( 9)         lt!MDATreq[CC,TRUE,0,0]
(10)          ut?ICONconf
(11)           ut!IDATreq[,0,0]
( 3)   lt!MDATreq[AK,TRUE,0,0]
( 4)    ut!IDATreq[,0,0]
( 5)     lt?MDATind[DT,FALSE,0,0]
( 6)      lt?MDATind[DT,FALSE,0,0]
( 7)       lt!MDATreq[AK,FALSE,0,0]
( 8)        ut!IDATreq[,0,0]
( 9)         lt?MDATind[DT,TRUE,0,0]
(10)          lt?MDATind[DT,TRUE,0,0]
(11)           lt!MDATreq[AK,TRUE,0,0]
(11)           lt?MDATind[DT,TRUE,0,0]
(10)          lt!MDATreq[AK,TRUE,0,0]
(11)           ut!IDATreq[,0,0]
( 7)       lt?MDATind[DT,FALSE,0,0]
( 8)        lt!MDATreq[AK,FALSE,0,0]
( 9)         ut!IDATreq[,0,0]
(10)          lt?MDATind[DT,TRUE,0,0]
(11)           lt?MDATind[DT,TRUE,0,0]
(11)           lt!MDATreq[AK,TRUE,0,0]
( 8)        lt?MDATind[DT,FALSE,0,0]
( 9)         lt!MDATreq[AK,FALSE,0,0]
(10)          ut!IDATreq[,0,0]
(11)           lt?MDATind[DT,TRUE,0,0]
( 9)         ut?IDISind
(10)          lt!MDATreq[AK,FALSE,0,0]
(11)           ut!ICONreq
( 9)         ut?IDISind
(10)          lt!MDATreq[AK,FALSE,0,0]
(11)           ut!ICONreq
(10)          ut!ICONreq
(11)           lt?MDATind[CR,FALSE,0,0]
( 6)      lt!MDATreq[AK,FALSE,0,0]
( 7)       ut!IDATreq[,0,0]
( 8)        lt?MDATind[DT,TRUE,0,0]
( 9)         lt?MDATind[DT,TRUE,0,0]
(10)          lt!MDATreq[AK,TRUE,0,0]
(11)           ut!IDATreq[,0,0]
(10)          lt?MDATind[DT,TRUE,0,0]
(11)           lt!MDATreq[AK,TRUE,0,0]
(11)           lt?MDATind[DT,TRUE,0,0]
( 9)         lt!MDATreq[AK,TRUE,0,0]
(10)          ut!IDATreq[,0,0]
(11)           lt?MDATind[DT,FALSE,0,0]
```

```
End of generation: 92-05-14 16:18:09
End of Testcases
Tree reduction ended : 02-04-93 12:20:20
Reduction time       : 00:00:01
Reduction factor     : 64% (375 -> 135)
Count of paths       : 43
```

E.7 REDUC-1: Verbindungsabbau für die Testkonfiguration 1

REDUC-1: This is a REDUCED version of file ABBAU.2
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 2
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : ENV, System-Process : Empfaenger
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-10 12:26:17

(1) lt!IDISreq
(2) lt!IDISreq
(2) ut?IDISind
(1) ut!IDATreq[,0,0]
(2) lt?IDATind[0,0]

End of generation: 92-05-10 12:26:30
End of Testcases
Tree reduction ended : 02-04-93 11:26:25
Reduction time : 00:00:00
Reduction factor : 0% (5 -> 5)
Count of paths : 3

E.8 REDUC-2: Verbindungsabbau für die Testkonfiguration 1

REDUC-2: This is a REDUCED version of file ABBAU.2
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 2
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : ENV, System-Process : Empfaenger
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-10 12:26:17

(1) lt!IDISreq
(2) lt!IDISreq
(2) ut?IDISind
(1) ut!IDATreq[,0,0]
(2) lt?IDATind[0,0]

End of generation: 92-05-10 12:26:30
End of Testcases
Tree reduction ended : 02-04-93 12:01:10
Reduction time : 00:00:00
Reduction factor : 0% (5 -> 5)
Count of paths : 3

E.9 REDUC-1: Verbindungsabbau für die Testkonfiguration 2

REDUC-1: This is a REDUCED version of file ABBAU.2
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 2
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-10 12:28:16

(1) lt!MDATreq[AK,FALSE,,]
(2) lt!MDATreq[AK,FALSE,,]
(2) lt!MDATreq[CC,FALSE,,]
(2) lt!MDATreq[DR,FALSE,,]
(2) ut!IDATreq[,0,0]
(1) lt!MDATreq[CC,,,]
(2) lt!MDATreq[AK,FALSE,,]

(2) lt!MDATreq[CC,,,]
(2) lt!MDATreq[DR,,,]
(2) ut!IDATreq[,0,0]
(1) lt!MDATreq[DR,,,]
(2) ut?IDISind
(1) ut!IDATreq[,0,0]
(2) lt?MDATind[DT,TRUE,0,0]

End of generation: 92-05-10 12:28:39
End of Testcases
Tree reduction ended : 02-04-93 11:32:22
Reduction time : 00:00:00
Reduction factor : 18% (17 -> 14)
Count of paths : 10

E.10 REDUC-2: Verbindungsabbau für die Testkonfiguration 2

REDUC-2: This is a REDUCED version of file ABBAU.2
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 2
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-10 12:28:16

(1) lt!MDATreq[AK,FALSE,,]
(2) lt!MDATreq[AK,FALSE,,]
(2) lt!MDATreq[CC,FALSE,,]
(2) lt!MDATreq[DR,FALSE,,]
(2) ut!IDATreq[,0,0]
(1) lt!MDATreq[CC,,,]
(2) lt!MDATreq[AK,FALSE,,]

(2) lt!MDATreq[CC,,,]
(2) lt!MDATreq[DR,,,]
(2) ut!IDATreq[,0,0]
(1) lt!MDATreq[DR,,,]
(2) ut?IDISind
(1) ut!IDATreq[,0,0]
(2) lt?MDATind[DT,TRUE,0,0]

End of generation: 92-05-10 12:28:39
End of Testcases
Tree reduction ended : 02-04-93 12:03:28
Reduction time : 00:00:00
Reduction factor : 18% (17 -> 14)
Count of paths : 10

E.11 REDUC-1: Verbindungsabbau für die Testkonfiguration 3

REDUC-1: This is a REDUCED version of file ABBAU.2
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 2
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-12 19:28:41

(1) lt!MDATreq[DR,,,]
(2) ut?IDISind
(1) ut!IDATreq[,0,0]
(2) lt?MDATind[DT,TRUE,0,0]

End of generation: 92-05-12 19:28:51
End of Testcases
Tree reduction ended : 02-04-93 11:48:33
Reduction time : 00:00:00
Reduction factor : 20% (5 -> 4)
Count of paths : 2

E.12 REDUC-2: Verbindungsabbau für die Testkonfiguration 3

REDUC-2: This is a REDUCED version of file ABBAU.2
TESTSEQUENCES OF SYSTEM INRES_Protokoll
MAX. DEPTH OF TREE : 2
REASONABLE ENVIRONMENT : YES
WEAK REASONABLE TIMER : NO
STRONG REASONABLE TIMER : YES
UPPER TESTER: Name : ut
ENV-Process : ENV, System-Process : Initiator
Signalroute : ISAP
LOWER TESTER: Name : lt
ENV-Process : Codierer_Empf, System-Process : MSAP_Mgr_Empf
REMOVE FROM NODECOMPARE:
Process: Codierer_Ini
Process: Codierer_Empf
Process: MSAP_Mgr_Ini
Process: MSAP_Mgr_Empf
Start of generation: 92-05-12 19:28:41

(1) lt!MDATreq[DR,,,]
(2) ut?IDISind
(1) ut!IDATreq[,0,0]
(2) lt?MDATind[DT,TRUE,0,0]

End of generation: 92-05-12 19:28:51
End of Testcases
Tree reduction ended : 02-04-93 12:20:09
Reduction time : 00:00:01
Reduction factor : 20% (5 -> 4)
Count of paths : 2