

Kurz-Tutorial Tau/SDT

Helmut Neukirchen
neukirchen@itm.mu-luebeck.de
Institut für Telematik
Medizinische Universität zu Lübeck

7. Mai 2001

1 Vorbereitungen

In der `.login`-Datei müssen die Pfade für Tau gesetzt sein.

Bei der Installation am Institut für Telematik geschieht dies durch `module load tau`. Nun entweder eine neue Shell starten, damit `.login` ausgeführt wird, oder `module load tau` von Hand eingeben.¹

(Da Tau standardmäßig den SUN C-Compiler verwendet, empfiehlt es sich außerdem, das Module `lang/sun_c_v5.2` zu verwenden. Der GCC ist aber ebenfalls möglich, allerdings muß SDT darüber gesondert informiert werden...)

Es empfiehlt sich, für die im Laufe des Tutorials erzeugten Dateien, ein eigenes Verzeichnis anzulegen (z.B. `SDL_Tutorial`) und in dieses zu wechseln.

2 Starten

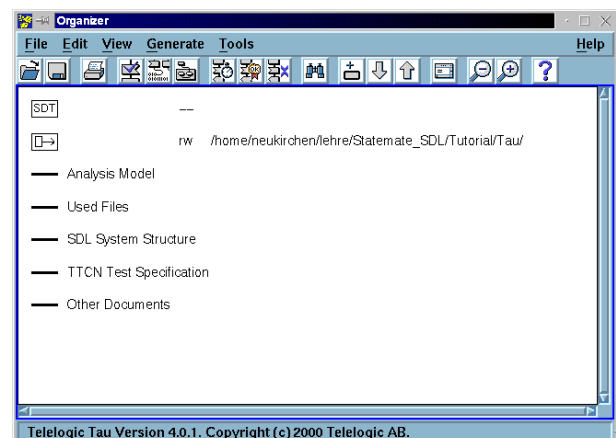
Das SDT-Tool wird mit `sdt` gestartet. Neben einem Begrüßungsfenster, das weggeklickt werden kann (*Continue*) erscheint das *Organizer*-Fenster.

Die schwarzen horizontalen Balken trennen verschiedene, "Kapitel" voneinander ab. Der Text rechts davon ist entsprechend soetwas wie eine "Überschrift". Da noch keine Dateien vorhanden sind, sind die "Kapitel" noch leer...

¹Tastatureingaben werden im weiteren mit der Schriftart *Courier* kenntlich gemacht. Alle Eingaben sowie ein gewisse graphische Rückkopplung werden außerdem nochmal am Rand aufgeführt. Auszuwählende Elemente der graphischen Oberfläche werden *kursiv* gekennzeichnet.

```
module load tau
```

```
sdt  
Continue
```



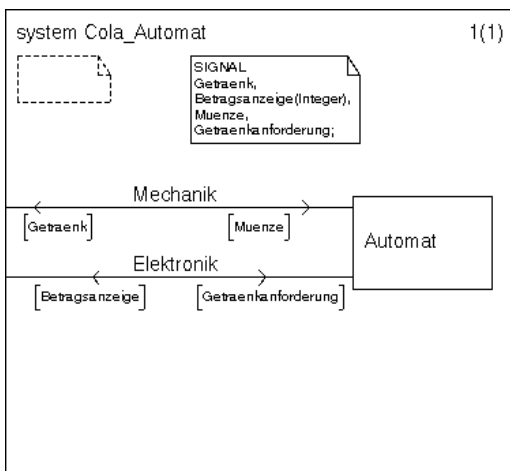
3 Editieren

3.1 Ein SDL-System hinzufügen

Im *Organizer* per Mausklick das ‘‘Kapitel’’ *SDL System Structure* per linker Maustaste selektieren und aus dem Hauptmenueintrag *Edit* den Punkt *Add New...* aufrufen. Im nun folgenden Dialog *Add New* von *Organizer* auf *SDL* umstellen, sowie als Diagrammart rechts daneben *System* einstellen Die Checkbox *Show in Editor* deaktivieren und als neuen Dokumentnamen *Cola_Automat* eingeben und auf *OK* klicken.

Daraufhin wird im *Organizer* zusatzlich eine Zeile mit *Cola_Automat* angezeigt. Die Angabe *[unconnected]* bedeutet, daB hierfür (noch) keine zugehoerige Datei existiert.

Im folgenden soll ein einfacher Cola-Automat, der mit Muenzen gefuettert wird und auf Tastendruck ein Kaltgetraenk ausgibt (sofern der eingeworfene Geldbetrag, der angezeigt wird, ausreicht) in *SDL* modelliert werden:



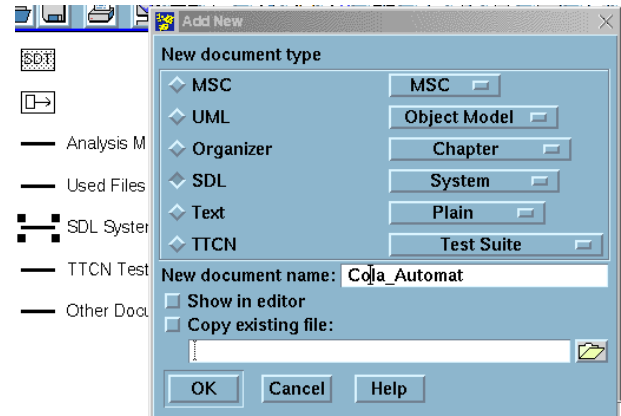
3.2 System editieren

Das oben abgebildete *SDL-System Cola_Automat* soll nun erstellt werden: Im *Organizer* auf die Zeile *Cola_Automat* doppelklicken. Im darauf erscheinenden *Edit*-Dialog, der dem vorhergehenden *Add New*-Dialog aehnelt, einfach *OK* klicken. Nun wird der *SDL-Editor* geoeffnet. Dieser zeigt ein leeres *System Cola_Automat* an.

3.2.1 Block hinzufuegen

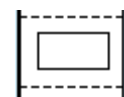
Als erstes soll ein *SDL-Block* hinzugefuegt werden. Hierzu aus der rechten Symbolleiste das Symbol (unten im Fenster werden die Namen der Symbole angezeigt) *Block Reference* mit der linken Maustaste auswaehlen und irgendwo in der Mitte der Arbeitsflaeche per Mausklick plazieren.

SDL System Structure selektieren
Edit→*Add New...*



SDL auswaehlen
System auswaehlen
Show in Editor deaktivieren
Cola_Automat
OK

Doppelklick auf *Cola_Automat*
OK



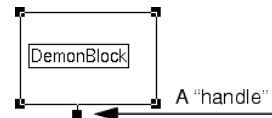
Block Reference auswaehlen
plazieren

Nun blinkt ein Cursor und wartet darauf, daß diesem *Block* ein Name gegeben wird: Automat. Ein Klick außerhalb des Textfeldes beendet die Eingabe.

Automat eingeben
Klick außerhalb

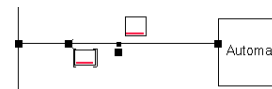
3.2.2 Kanal hinzufügen

Der *Block Automat* soll eine Verbindung mit dem *Environment* (der Außenwelt, die durch den rechteckigen Rahmen um das *System* herum begrenzt ist.) bekommen. Hierzu besitzt das *Block*-Symbol ein "Handle", das erscheint, sobald ein Symbol per Mausklick angewählt wurde. Ausgehend von diesem "Handle" nun mit gedrückter linker Maustaste einen *Kanal* zum *Environment* ziehen, so daß z.B. der linke Rand des Rahmens genau berührt wird. Durch Loslassen der Maustaste wird der Endpunkt des *Kanals* festgelegt.



Block selektieren, so daß "Handle" erscheint
"Handle" mit gedrückter linker Maustaste nach links auf die Umrandungslinie ziehen und loslassen.

Es erscheinen zunächst zwei neue Textfelder ober- und unterhalb des *Kanals*. Durch einen Klick auf die rechte Maustaste erscheint ein Kontextmenu, aus dem der Eintrag *Bidirect* ausgewählt wird. Es erscheint ein drittes Textfeld.



Klick auf die rechte Maustaste
Bidirect auswählen

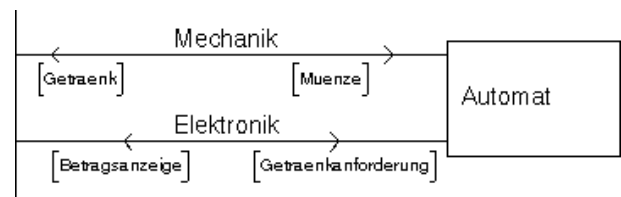
In das obere Textfeld den Namen des *Kanals* eingeben: Textfeld mit der linken Maustaste auswählen und anschließend *Mechanik* eingeben und auf das untere linke Textfeld klicken. Dort werden die Namen der *Signale*, die über diesen *Kanal* zum *Environment* geschickt werden, eingegeben: *Getraenk*. Zuletzt noch in das rechte Textfeld klicken und anschließend *Muenze* eingeben.

oberes Textfeld auswählen
Mechanik eingeben
auf das untere linke Textfeld klicken
Getraenk eingeben
auf das untere rechte Textfeld klicken, *Muenze* eingeben

Analog muß ein zweiter *Kanal Elektronik*, der parallel zum *Environment* verläuft, erstellt werden. Zum *Environment* läuft das *Signal Betragsanzeige*, in der Gegenrichtung das *Signal Getraenkanforderung*.

Zweiten *Kanal Elektronik* anlegen, *Signale*: *Betragsanzeige* und *Getraenkanforderung*

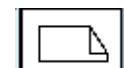
Nach einem Klick irgendwo auf den weißen Arbeitsbereich können noch optische Schönheitskorrekturen vorgenommen werden: Die *Kanäle* können mit der Maus verschoben werden, an der linken Seite können Textfelder "angepackt" (die Mauszeigerform ändert sich) und verschoben werden. Textfelder können nachträglich editiert oder per Eingabetaste umgebrochen werden.



3.2.3 Signaldeklaration

Die ausgetauschten *Signale* müssen noch genauer bzgl. ihres Typs deklariert werden. Dies geschieht innerhalb eines *Text*-Symbols:

Aus der Symbolleiste ganz oben *Text* auswählen und z.B. in die Mitte der oberen Arbeitsfläche ziehen. In diesem Symbol blinkt nun ein Cursor und die *Signaldeklaration* kann eingegeben werden:



Text auswählen
plazieren

SIGNAL Getraenk, Betrags²anzeige(Integer), Muenze, Getraenkanforderung;

Ein Klick "in's Leere" beendet die Eingabe.

Das eingegebene System sollte nun dem in Abschnitt 3.1 gezeigten SDL System Cola_Automat ähneln.

3.2.4 Speichern

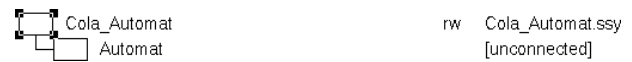
Durch Auswahl von *Save* im Menüpunkt *File* und Bestätigen mit *OK* kann das System gespeichert werden. Mit *Close Diagram* im Menüpunkt *File* kann der Editor beendet werden.

Im *Organizer*-Fenster ist nun unter dem System *Cola_Automat* ein neuer Eintrag hinzugekommen, der den neu angelegten *Block Automat* repräsentiert.

SIGNAL Getraenk, Betragsanzeige(Integer), Muenze, Getraenkanforderung; eingeben

"in's Leere" klicken

File→*Save*
OK
File→*Close Diagram*



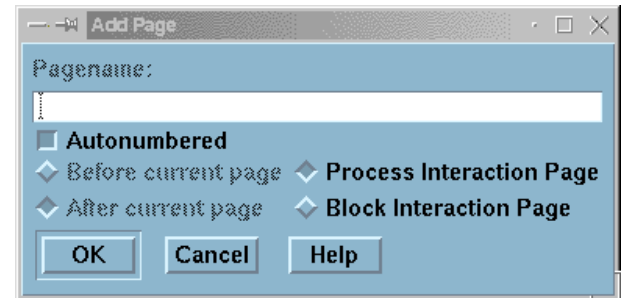
3.3 Einen SDL-Block editieren

Nun soll der *Block Automat* editiert werden: Ein Doppelklick auf die Zeile *Automat* im *Organizer*-Fenster öffnet den *Edit*-Dialog, der mit *OK* bestätigt wird. Ein *Add Page*-Dialog erscheint, der ebenfalls mit *OK* bestätigt wird. Daraufhin erscheint der *SDL Editor* für den *Block Automat*.

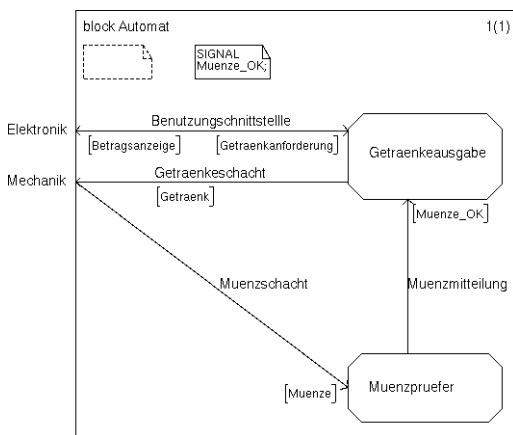
Doppelklick auf *Automat* im *Organizer*

In diesem Abschnitt soll der nachfolgende *SDL-Block* erstellt werden:

OK

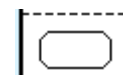


OK



3.3.1 Prozess hinzufügen

Zunächst wird das Symbol *Process Reference* ausgewählt und in die obere Hälfte der Arbeitsfläche plaziert. Dieser neue *Prozess* bekommt den Namen *Getraenkeausgabe*.



Process Reference auswählen
 plazieren

Getraenkeausgabe

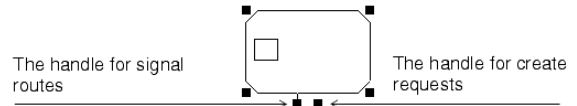
²Der Autor, der vom Niederrhein stammt, bevorzugt an dieser Stelle ein Binnen-S – die eher norddeutsch angehauchten lassen dies weg. So oder so solltet ihr aber im weiteren Verlauf dabei konsequent sein!

Der zweite *Prozess* soll Muenzpruefer heißen und kann auf die gleiche Weise darunter plaziert werden.

Prozess Muenzpruefer anlegen und plazieren

3.3.2 Signalrouten hinzufügen

Prozesse besitzen – im Gegensatz zu *Blöcken* – zwei “Handles”. Das rechte “Handle” kann dazu benutzt werden, dynamisch *Prozessinstanzen* zu erzeugen – wir werden hiervon jedoch keinen Gebrauch machen. Mit dem linken “Handle” können – wie gewohnt – Verbindungen gezogen werden (auf dieser Ebene heißen sie jedoch *Signalrouten* und stellen eine Verfeinerung von *Kanälen* dar.).



Die erste *Signalroute* soll von der *Getraenkeausgabe* zum *Environment* gehen: *Getraenkeausgabe* selektieren und ausgehend von diesem “Handle” nun mit gedrückter linker Maustaste einen *Kanal* zum *Environment* ziehen, so daß der z.B. linke Rand des Rahmens genau berührt wird. Durch Loslassen der Maustaste wird der Endpunkt der *Signalroute* festgelegt.

Getraenkeausgabe selektieren

“Handle” mit gedrückter linker Maustaste nach links auf die Umrandungslinie ziehen und loslassen.

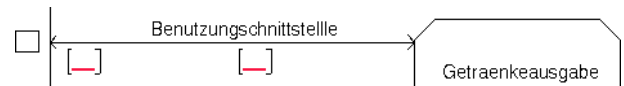
Durch einen Klick auf die rechte Maustaste erscheint ein Kontextmenu, aus dem der Eintrag *Bidirect* ausgewählt wird. Es erscheint ein drittes Textfeld.

Klick auf die rechte Maustaste

Bidirect auswählen

Nun kann der Name dieser *Signalroute* eingegeben werden: Oberes Textfeld anklicken und Benutzungsschnittstelle eingeben.

Oberes Textfeld anklicken, Benutzungsschnittstelle eingeben

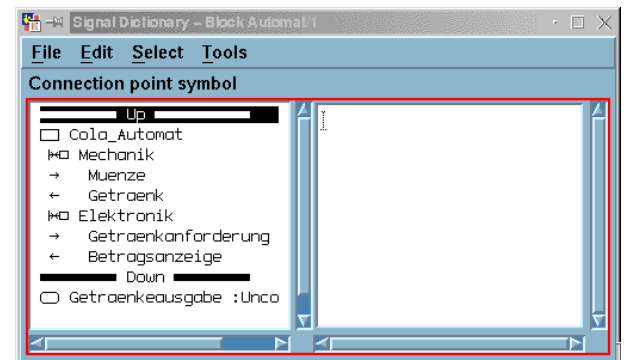


Außerhalb der Umrandung des *Environments* befindet sich ein neues Textfeld, das den *Kanal*, über den diese *Signalroute* nach außen geführt wird, angibt. Dieser kann entweder per Hand oder über das *Signal Dictionary* eingegeben werden. Letzteres geht wie folgt: Das *Signal Dictionary* ist über den Menüpunkt *Windows*→*Signal Dictionary* zu erreichen.

Windows→*Signal Dictionary*

Nachdem im *SDL Editorfenster* das Textfeld für den *Kanal* ausgewählt wurde, werden im *Signal Dictionary* alle bisherigen Signaldefinitionen angezeigt. Ein Doppelklick auf *Elektronik* fügt diesen *Kanalnamen* ein.

im *SDL Editorfenster* das Textfeld für den *Kanal* anklicken



Im linken unteren Textfeld der *Signalroute* müssen nun noch die *Signale*, die über diese Route nach außen übertragen werden, eingegeben werden: Dieses Feld selektieren und ein anschließender Doppelklick auf *Betragsanzeige* im *Signal Dictionary*-Fenster spezifiziert, daß über diese *Signalroute* in dieser Richtung nur der Betrag ausgegeben wird. In der Gegenrichtung soll das *Signal* *Getraenkanforderung* empfangen werden: Das rechte untere Textfeld selektieren und Doppelklick auf *Getraenkanforderung* im *Signal Dictionary*-Fenster.

Doppelklick auf *Elektronik*

im *SDL Editorfenster* das linke untere Textfeld per Mausklick auswählen

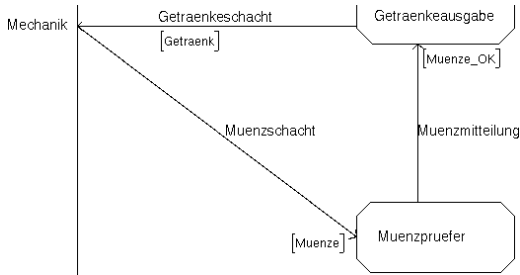
Doppelklick auf *Betragsanzeige*

im *SDL Editorfenster* das rechte untere Textfeld per Mausklick auswählen

Doppelklick auf *Getraenkanforderung*

Analog dazu müssen die verbleibenden *Signalrouten Getraenkeschacht, Muenzschacht* und *Muenzmitteilung* wie unten abgebildet angelegt werden.

Signalrouten Getraenkeschacht, Muenzschacht und *Muenzmitteilung* analog anlegen



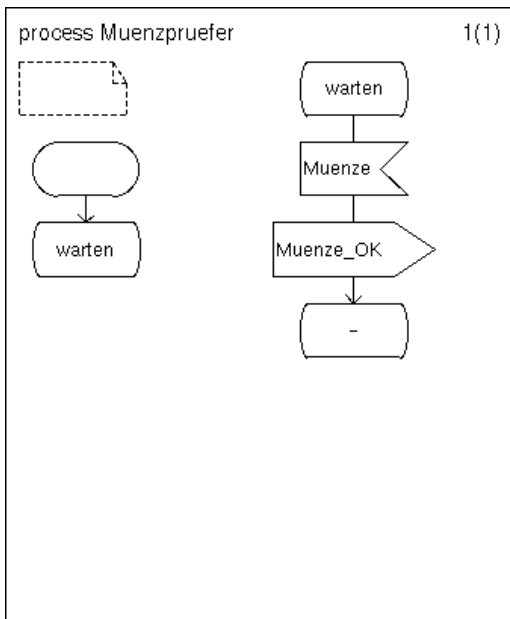
3.4 Einen SDL-Prozess editieren

Da die strukturellen Eigenschaften des Automaten nun festgelegt sind, soll nun dessen Verhalten spezifiziert werden. Durch das Hinzufügen der beiden *Prozesse Getraenkeausgabe* und *Muenzpruefer* sind im *Organizer*-Fenster die zugehörigen Einträge ebenfalls aufgeführt.

Dort öffnet ein Doppelklick auf *Muenzpruefer* den *Edit*-Dialog, der mit *OK* bestätigt wird. Der *Add Page*-Dialog erscheint und kann ebenfalls mit *OK* bestätigt werden. Daraufhin erscheint der *SDL Editor* für den *Prozess Muenzpruefer*.

Doppelklick auf *Muenzpruefer* im *Organizer*
OK
OK

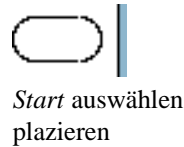
In diesem Abschnitt soll zunächst der nachfolgende *SDL-Prozess* erstellt werden:



3.4.1 Flußdiagramm hinzufügen

Die Flußdiagramme innerhalb von *Prozessen* besitzen ein *Start*-Symbol für den Startzustand des *SDL-Automaten*.

Start-Symbol mit der linken Maustaste aus der Symbolleiste rechts auswählen und im oberen Bereich der Arbeitsfläche per Mausklick plazieren. Einen Namen bekommt dieser Zustand nicht, deshalb wird gleich das nächste Symbol angehängt:



Ohne zwischenzeitlich mit der Maus herumzuklicken auf das State-Symbol doppelklicken. Dieses Symbol wird automatisch an das zuletzt selektierte Symbol angehängt. Nun kann der Namen dieses neuen Zustands eingegeben werden: warten. Da diese Transition (d.h. Baum von Symbolen mit State-Symbol an der Wurzel und an den Blättern) bereits jetzt zu Ende sein soll, einfach mit der Maus irgendwo in den weißen Bereich der Arbeitsfläche klicken.

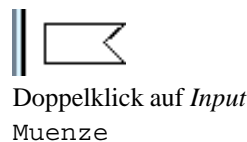


Für die nächste Transition erneut auf das State-Symbol in der Symbolleiste doppelklicken. Ein State-Symbol erscheint am oberen Rand der Arbeitsfläche. Über Tastatur kann der Namen dieses Zustands eingegeben werden: warten. (Falls einem die vom *SDL-Editor* gewählte Position nicht gefällt, kann diese natürlich noch per Maus geändert werden.)

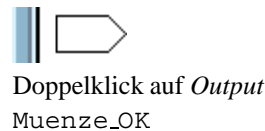
irgendwo in den weißen Bereich der Arbeitsfläche klicken



An das selektierte State-Symbol soll nun ein Input-Symbol angehängt werden: Ein Doppelklick hierauf in der Symbolleiste sollte dies zur Folge haben. Es soll auf das *Signal Muenze* gewartet werden: Bei in diesem Symbol blinkendem Cursor Muenze eingeben.



Unter der mysteriösen Annahme, daß die Gültigkeit der Münze auf eine mysteriöse Weise geprüft wird, soll die Gültigkeit der Münze mitgeteilt werden: Doppelklick auf das *Output*-Symbol und *Muenze_OK* eingeben.

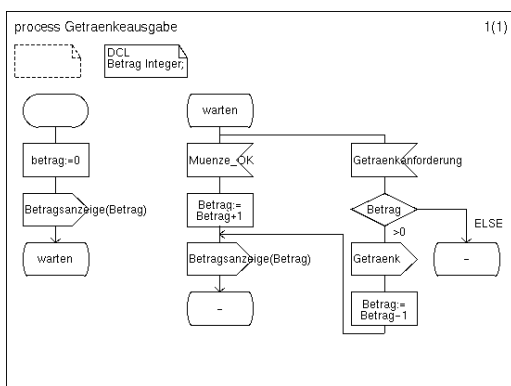


Im Anschluß soll wieder in den Ausgangszustand übergegangen werden. Da wir diesmal schreibfaul sind, wird nach dem obligatorischen Doppelklick auf das State-Symbol in der Symbolleiste einfach nur - eingegeben, was eine Abkürzung für den Zustand ist, mit dem die *Transition* begonnen hat.



Fertig ist der Münzprüfer! Mit *Save* aus dem *File*-Menü kann das Diagramm wie gewohnt gespeichert werden.

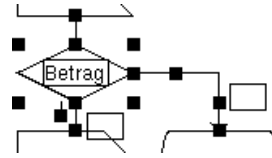
File→Save



Bitte selbständig eingeben und speichern.

Das Diagramm für *Getraenkeausgabe* kriegt ihr jetzt doch alleine hin, oder? (*Decision* ist vielleicht ein wenig tricky: Um die

Textfelder für die Bedingungen zu bekommen, muß nochmal auf das Symbol und anschließend in das dann erscheinende Textfeld geklickt werden.)



4 Analysieren

Um zu testen, ob die Spezifikation syntaktisch (und grob semantisch) korrekt ist, soll nun der *Analyzer* angeworfen werden. (Syntaktische Fehler werden – wie euch sicherlich schon aufgefallen ist – im übrigen bereits bei der Eingabe rot unterstrichen.)

Im *Organizer* muß das zu analysierende *System* angeklickt werden, d.h. das *System Cola_Automat*. Der *Analyzer* versteckt sich im Menü *Generate* unter dem Punkt *Analyze...*. Der *Analyze SDL*-Dialog erscheint.

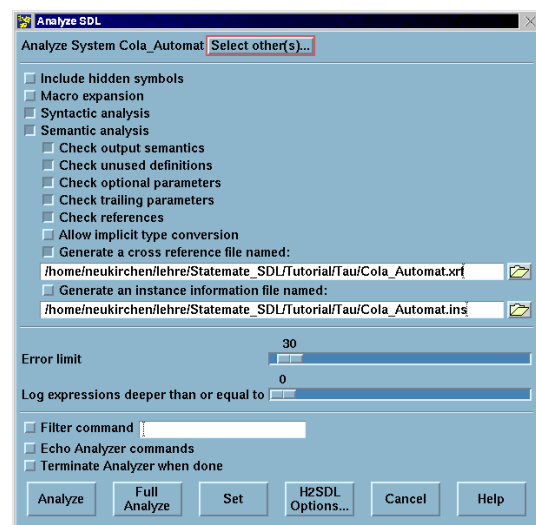
Mit den vielen Optionen kann eingestellt werden, worauf bei der Analyse geachtet werden soll. So kann z.B. festgestellt werden, ob Signale verschickt werden, die nie empfangen werden und umgekehrt oder es kann geprüft werden, ob Signale deklariert wurden, die nie benutzt werden. Die Optionen sollten wie nebenstehend gesetzt sein.

Falls mal ein halbfertiges System gecheckt werden soll, empfiehlt es sich, die semantische Analyse komplett abzustellen, da sich hier sonst die Fehlermeldungen häufen.

Ein Klick auf *Analyze* startet die Analyse. Wenn alles richtig gemacht wurde, sollte nach einiger Zeit *Analyzer done: No errors and no warnings* in der unteren Zeile des *Organizers* erscheinen.

Im Fehlerfall erscheint das Fenster *Organizer Log* mit Fehlermeldungen. (Unter *Tools*→*Organizer Log* kann dieses per Hand aufgerufen werden.) Um die angezeigten Fehler in den SDL-Diagrammen zu lokalisieren: Zeilen der Fehlermeldung wie nebenstehend per Maus markieren und *Tools*→*Show Error* aus dem Menü auswählen.

System Cola_Automat anklicken
Generate→*Analyze...*



Analyze

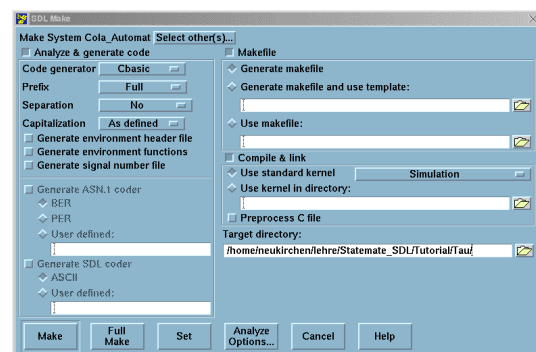
```
#SDTREF[SDL_C:\sd31\work\demongame\DemonGame.ssy(1),132(17,52),2,8]
ERROR 312 Syntax error in rule SIGNALLIST, symbol Name found but one of the
following expected:
; comment
Result Endgame;
?
```

5 Simulieren

Das erstellte SDL-Modell soll nun simuliert werden. Hierzu muß im *Organizer* das zu simulierende *System* angeklickt werden, d.h. das *System Cola_Automat*. Der *Simulator* versteckt sich im Menü *Generate* unter dem Punkt *Make...*. Der *SDL Make*-Dialog erscheint.

Die Optionen sollten wie abgebildet eingestellt sein. Insbesondere muß in der Abteilung *Compile & Link* die Option *Use standard Kernel* eingestellt und *Simulation* ausgewählt sein. (Für GCC-BenutzerInnen: *gcc-Simulation*). Die Übersetzung

System Cola_Automat anklicken
Generate→*Make...*



des SDL-Systems in ein ausführbares Programm erfolgt durch Klicken auf *Make*.

Make

(Prinzipiell könnte SDL natürlich auch interpretiert statt kompiliert werden, aber gerade bei großen Systemen, die nicht nur interaktiv simuliert sondern auch automatisch validiert (s.u.) werden sollen, wäre dies zu langsam!)

Im *Organizer Log* sollten die Meldungen des C-Compilers erscheinen. Falls es dort Fehlermeldungen gibt, hilft vielleicht *Full Make* statt *Make*, da dann alle Dateien neu übersetzt werden.

5.1 Simulator aufrufen

Die graphische Oberfläche des Simulators, der per *Make* erzeugte ausführbare SDL-Systeme laden kann, wird im *Organizer* mit *Simulator UI* aus dem Submenü *SDL* des *Tools*-Menüs aufgerufen.

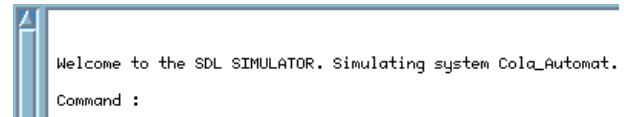
Tools→*SDL*→*Simulator UI*

Das Fenster *SDL Simulator UI* erscheint. Zunächst muß das soeben erzeugte, ausführbare SDL-Systeme geladen werden. Dies geschieht per *Open...* im Menü *File* des Simulators und der anschließenden Auswahl (Fileselektor) der Datei *Cola_Automat_smd.sct*.

File→*Open...*

Cola_Automat_smd.sct auswählen, *OK*

Nun wartet der Simulator auf Kommandos.



5.1.1 Anzeigoptionen setzen

Zunächst sagen wir dem Simulator, was wir alles von ihm angezeigt bekommen wollen.

Wir wollen im Textfenster des Simulators alle Informationen zum gerade ausgeführten Einzelschritt: *Text Level: Set* im Menü *Trace*. Darauf erscheint ein *Select*-Dialog, in dem angegeben werden könnte, auf welche SDL-Teilsysteme sich die nachfolgende Einstellung beziehen soll. Da alle gewünscht sind, kann einfach – ohne etwas auszuwählen – auf *OK* geklickt werden. (Manchmal legt die Simulatoroberfläche dabei eine Gedenkpause ein (Mauszeiger wird zum Fragezeichen) – der Simulator handelt mit dem ausführbaren SDL-System aus, was es gleich anzeigen soll: das dauert was...) Im darauffolgenden *Select*-Dialog wählen wir die Stufe *6 /* All SDL actions + Result + Parameters */* und klicken *OK*.

Trace→*Text Level: Set*

OK

6 / All SDL actions + Result + Parameters */* auswählen und *OK*

Da kein Mensch das komplette SDL-System im Kopf hat, wollen wir während des Simulierens wissen, wo wir uns gerade im System befinden: *SDL Level: Set* im Menü *Trace*. Darauf erscheint – wie zuvor – der *Select*-Dialog, in dem wir ebenfalls einfach *OK* klicken. Im darauffolgenden *Select*-Dialog wählen wir die Stufe *1 /* Show next symbol when entering monitor */* und klicken *OK*. Im Menü *Show* auf *Next Symbol* klicken und der *SDL-Editor*

Trace→*SDL Level: Set*

OK

1 / Show next symbol when entering monitor */* auswählen und *OK*

Show→*Next Symbol*

wird gestartet.

5.2 Das System ausführen

Nun kann schrittweise das SDL-System ausgeführt werden. Es stehen dabei verschiedene Granularitäten zur Verfügung.

5.2.1 Symbolweise ausführen

Zunächst wollen wir symbolweise durch das System steppen. Dies geschieht mit dem Button *Symbol* in der Abteilung *Execute*.

Im *SDL-Editor* wird nun das *Symbol* angezeigt, das zur Ausführung ansteht, und im Textfenster des Simulators wird der Beginn der *Transition* angezeigt. (Evtl. wird nicht der Prozess *Getraenkeausgabe* – wie abgebildet – sondern der Prozess *Muenzpruefer* zuerst ausgeführt.)

Nun solange symbolweise das System ausführen, bis die aktuelle *Transition* beendet ist, d.h. der Zustand *warten* eingenommen wird, was im Textfenster wie nebenstehend angezeigt wird. (Der *SDL-Editor* zeigt immer das nächste auszuführende *Symbol* an – nicht das aktuelle!) Die Sternchen umfassen jeweils eine *Transition* (die aus mehreren *Symbolen* besteht) wie eine eckige Klammer.

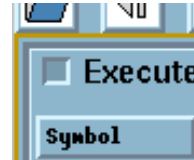
Als nächstes durchlaufen wir auf dieselbe Weise die *Starttransition* des nächsten Prozesses.

5.2.2 Signale senden

Irgendwann meldet der Simulator, daß keine weiteren *Transitionen* mehr zum Ausführen da sind. Das SDL-System wartet nun auf eine Eingabe, z.B. den Einwurf einer Münze.

Dies geschieht in der Abteilung *Send Signal*. Da wir uns nicht gemerkt haben, an wen wir die *Signale* schicken sollen, verwenden wir den Button *Send Via* in der Abteilung *Send Signal*. Darauf erscheint – mal wieder – der *Select-Dialog*, in dem das *Signal Muenze* z.B. per Doppelklick ausgewählt wird. Nun muß noch der *Kanal* angegeben werden, über den dieses *Signal* geschickt werden soll. Da *Muenze* nur über einen *Kanal* geschickt werden kann, reicht *OK*.

Da nun der nötige Stimulus vorliegt, kann das System wieder weiter ausgeführt werden.



Symbol

```
*** TRANSITION START
*   Pid   : Getraenkeausgabe:1
*   State : start state
*   Now   : 0.0000
```

Symbol bis zum Ende der *Transition*

```
Command : Step-Symbol
*** NEXTSTATE warten
```

Symbol bis zum Ende der *Transition*

```
Command : Step-Symbol
No process instance scheduled for a transition
```



Send Via

Doppelklick auf *Muenze*

OK

5.2.3 Noch ein paar Features des Simulators

Wer will, kann nun noch weiter symbolweise durch das System laufen und z.B. die Reaktion auf das *Signal Getraenkanforderung* studieren. Schneller geht's, indem transitionsweise durch das System gesteppt wird. Dafür ist das aber auch nicht so spannend, da das System im Zweifel nur in *warten*-Zuständen stehenbleibt und der Simulator die nächste *Transition* noch nicht anzeigen kann, da diese von der nächsten Eingabe, die noch nicht feststeht, abhängt.



Wenn man noch mehr Interna des SDL-Systems sehen will, empfiehlt sich das *View*-Menü:

In das *Watch Window* lassen sich per *Watch*→*Add...* Variablennamen eintragen, deren Wert in jedem Simulationsschritt aktualisiert angezeigt wird (z.B. *Betrag*, *OK*).

View→*Watch Window*
Watch→*Add...*
Betrag, *OK*

Im *Command Window* lassen sich die laufend aktualisierten Ergebnisse bestimmter Simulator-Befehle anzeigen. Standardmäßig sind dort die Listen für *Ready-Queue* und *Process* aufgeführt (evtl. Fenster größer ziehen!). Per *Command*→*Add Command...* läßt sich z.B. der Befehl *List-Input-Port* dauerhaft anzeigen, so die Input-Queue des aktuellen Prozesses beobachtet werden kann.

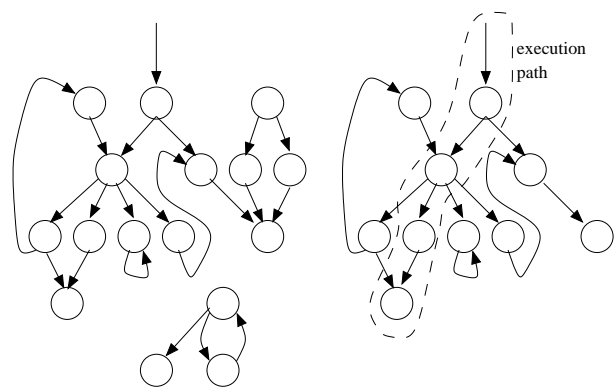
View→*Command Window*
Command→*Add Command...*
List-Input-Port, *OK*

Kurz bevor Du die Lust am Simulator verlierst, solltest Du diesen – mit welchem Menüpunkt bloß? – beenden. Falls Du das *Command Window* oder *Watch Window* modifiziert hast, kannst Du beim Beenden noch sagen, ob diese Änderungen beim nächsten Mal übernommen werden sollen oder nicht. Da Du den Simulator noch nicht so gut kennst, sag' im Zweifel lieber *Discard*!

6 Validieren

Der Validator ist dem Simulator recht ähnlich. Das Konzept der Validierung sieht vor, daß alle verschiedenen Zustände, die das SDL-System annehmen kann, herausgefunden werden, um festzustellen, ob es irgendwo Probleme geben könnte (z.B. es ist ein Zustand erreichbar, in dem ein Signal gesendet werden soll, das in dieser Situation von niemandem empfangen werden kann).

Fleissige können dies per Hand mit dem Simulator machen, indem alle Permutationen der jeweils möglichen Eingaben ausprobiert werden, ein wenig einfacher geht's mit dem Validator. Dieser ermittelt – ausgehend vom Startzustand – alle möglichen Folgezustände, indem ggf. die nötigen Eingaben gesendet werden, und setzt diese Suche rekursiv fort, um so den *Zustandsraum* bzw. den *Erreichbarkeitsgraphen* (aus dem sich alle möglichen *Ausführungspfade* ergeben) des Systems zu ermitteln.



(a) State diagram

(b) Reachability graph

Der Validator kann direkt im *Organizer* per Buttonleiste aufge-

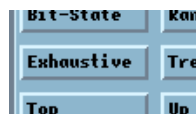
rufen, werden, wobei darauf zu achten ist, daß zu diesem Zeitpunkt das *System Cola Automat* ausgewählt ist. Nachdem – wie beim Simulator – ein ausführbares SDL-System erzeugt wurde, erscheint das Fenster *SDL Validator UI*.

System Cola Automat auswählen



6.1 Vollständige Validierung

Zunächst soll der Validator versuchen, den kompletten *Zustandsraum* des Automaten zu erkunden. Dies geschieht über den Button *Exhaustive* in der Abteilung *Explore*.



Exhaustive

Darauf rattert der Validator los und beendet den Versuch der vollständigen Validierung mit einem *Report Viewer*-Fenster, in dem – falls Fehler gefunden wurden – der genaue Fehlertyp angezeigt wird (Doppelklick hilft dann weiter). Außerdem erscheint eine Statistik im Textfeld: Die Suchtiefe betrug 100, d.h. die maximale Länge eines untersuchten möglichen *Ausführungspfades* war 100 Zustände. In 3 Fällen wurde diese maximale Länge erreicht, d.h. 3 *Ausführungspfade* konnten nicht bis zu Ende geführt werden, dennoch wurde alle SDL-Symbole mindestens einmal besucht (100% coverage).

```

** Starting exhaustive exploration **
Search depth : 100

** Exhaustive exploration statistics **
No of reports: 0
Generated states: 319
Truncated paths: 3.
Unique system states: 257.
Size of hash table: 100000 (400000 bytes)
Current depth: -1
Max depth: 100
Min state size: 56
Max state size: 112
Symbol coverage : 100.00

```

Warum wurden einige *Ausführungspfade* nicht bis zu Ende ausgeführt? Die Tatsache, daß dennoch 100% Symbolüberdeckung vorliegt, deutet darauf hin, daß es sich hierbei um schleifenartige Pfade handelt. Solche Pfade sind möglich, da ja z.B. beliebig oft hintereinander Münzen in den Automaten geworfen werden, was dazu führt, daß die Variable *Betrag* hochgezählt wird und der Pfad der bisher besuchten Zustände, die sich jeweils in der Variablen *Betrag* unterscheiden, länger wird. Der *Zustandsraum* des Systems ist daher unendlich groß und kann nie vollständig validiert werden!

6.2 Validieren per Navigation

In solchen Fällen, in denen eine vollständige Validierung nicht möglich ist, bleiben verschiedene Möglichkeiten, zumindest Teile des Systems gezielt zu besuchen. Eine Möglichkeit ist der *Navigator*, der – vom aktuellen Zustand ausgehend – alle erreichbaren Folgezustände anzeigt und die Auswahl des zu besuchenden Folgezustands ermöglicht. (Im Gegensatz zum Simulator, der auch symbolweise durch das System laufen kann, erlaubt der Validator nur, von einer Transition zur nächsten zu laufen.)

Um auch noch zu einem späteren Zeitpunkt nachvollziehen zu können, welcher *Ausführungspfad* dabei durchlaufen wurde, kann *Toggle MSC Trace* aus dem *Command*-Menü aufgerufen werden. Dadurch werden die im folgenden auftretenden Ereignisse “mitgeschnitten”.

Command → *Toggle MSC Trace*

Das Fenster *MSC Editor* erscheint und zeigt einen *Message Sequence Chart* mit dem Namen *Validator Trace* an. Da die Namen

der SDL-Prozesse relativ lang sind, überlappen diese sich teilweise. Um dies zu ändern, sollen die senkrechten (Zeit-)Achsen verschoben werden, damit der *MSC* wie nebenstehend aussieht:

In die weiße Arbeitsfläche klicken, damit nichts mehr selektiert ist. Nun die Achse des Prozesses *Muenzpruefer* mit gedrückter linker Maustaste ein wenig nach rechts schieben und dort loslassen. Das gleiche Spiel wiederholen wir dann mit der mittleren Achse.

Der eigentliche *Navigator* versteckt sich hinter einem Klick auf den Button *Navigator* in der Abteilung *Explore*.

Das *Navigator*-Fenster erscheint. Dort wird im Kasten *Next 1* der nächste erreichbare Zustand angezeigt. Zunächst kommt nur die Starttransition in Frage. Nach einem Doppelklick auf diesen Kasten erscheint der nächste erreichbare Zustand. Dies ist der Startzustand des zweiten Prozesses. Im *MSC Editor* erscheinen nun zeitgleich die Nachrichten, die in jedem Schritt ausgetauscht werden.

Nachdem der nächste Zustand ebenfalls per Doppelklick besucht wurde, stehen zwei mögliche *Signale*, die vom *Environment* zum System geschickt werden können, zur Auswahl. Anhand dieser Eingabe entscheidet sich, welcher Zustand als nächstes angenommen wird.

Da eine sinnvolle Benutzung des Cola-Automaten mitgeschnitten werden soll, soll zunächst eine Münze eingeworfen werden. D.h. es ist der Zustand auszuwählen, in dem das *Signal Muenze* an den Empfänger *Muenzpruefer* geschickt wird.

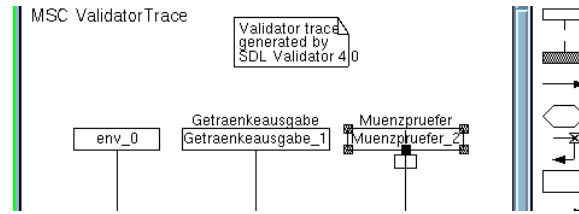
(Falls man sich verklickt hat bzw. sich später noch entscheidet, doch einen anderen Pfad einzuschlagen, kann mit dem Kasten *Up 1* auch zurück navigiert werden.)

Die nächsten beiden Zustände sollen ebenfalls angenommen werden, damit der Münzeinwurf in der Getränkeausgabe auch wahrgenommen wird.

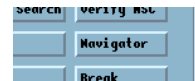
Nun soll ein Getränk angefordert werden: D.h. es ist der Zustand auszuwählen, in dem das *Signal Getraenkanforderung* an den Empfänger *Getraenkeausgabe* geschickt wird. Die Getränkeausgabe soll dieses Signal nun konsumieren.

An dieser Stelle beenden wir die Validierung per *Navigator*. Zunächst soll der erzeugte *MSC* unter dem Namen *TraceLeadingToSuccess.msc* gespeichert werden. Dies geschieht im *MSC Editor* per *Save as...* im *File*-Menü.

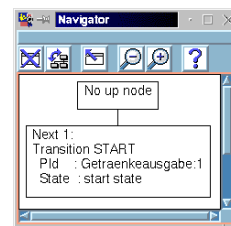
Die Aufzeichnung dieser *MSCs* kann nun beendet werden: *Toggle MSC Trace* aus dem *Command*-Menü. Das *MSC Editor*-Fenster verschwindet.



In die weiße Arbeitsfläche klicken
Muenzpruefer nach rechts ziehen
Getraenkeausgabe nach rechts ziehen



Navigator



Doppelklick auf Kasten *Next 1*
Doppelklick auf Kasten *Next 1*

Doppelklick auf Kasten mit *Signal Muenze* und Empfänger *Muenzpruefer*

Doppelklick auf Kasten *Next 1*
Doppelklick auf Kasten *Next 1*

Doppelklick auf Kasten mit *Signal Getraenkanforderung* und Empfänger *Getraenkeausgabe*

Doppelklick auf Kasten *Next 1*

File→*Save as...*, *TraceLeadingToSuccess.msc*, *OK*

Command→*Toggle MSC Trace*

6.3 Erfüllbarkeit von MSCs validieren

Wir haben nun mit Hilfe des Validators sichergestellt, daß es in unserem Modell einen Pfad gibt, der es erlaubt, nach Münzeinwurf ein Getränk am Automaten zu ziehen. Als letztes soll noch überprüft werden, ob es möglich ist, nur einmal zu bezahlen, aber zweimal hintereinander ein Getränk zu ziehen.

6.3.1 MSCs editieren

Hierzu fügen wir im *MSC Editor* noch eine zweite Getränkanforderung samt (fälschlicher) Getränk Ausgabe hinzu.

Zunächst muß der *MSC Editor* wieder gestartet werden. Im *Organizer*-Menü *Tools* aus dem Submenü *Editors* den Punkt *MSC Editor* aufrufen. Mit *File*→*Open* den *MSC TraceLeadingToSuccess.msc* laden.

Nun mit gedrückter *Shift*-Taste die beiden Pfeile der Nachrichten *Getraenkanforderung* und *Getraenk* mit der Maus selektieren und aus dem Kontextmenü, das per rechter Maustaste aufgerufen werden kann, *Copy* wählen. Nun die Auswahl deselektieren, indem in den weißen Arbeitsbereich geklickt wird. Anschließend aus dem Kontextmenü *Paste* auswählen, so daß die beiden Nachrichtenpfeile am Mauszeiger "kleben". Diese sollen nun einfach parallel zu den ursprünglichen Nachrichtenpfeilen unter das letzte *Warten*-Symbol plaziert werden.

Dieser neue *MSC* soll unter dem Namen *TraceLeadingToFail.msc* gespeichert werden.

Der *MSC Editor* kann nun mit *Exit* im Menü *File* beendet werden. Im *Organizer* wurde nun eine *Untitled*-Zeile für diesen *MSC* angelegt. Da diese Anzeige nur irritiert, kann diese markiert und per *Remove...* im Menü *Edit* und anschließender Bestätigung aus der *Organizer*-Anzeige entfernt werden.

6.3.2 MSCs validieren

Für die beiden gespeicherten *MSCs* kann nun mit dem Validator überprüft werden, ob es im *SDL*-System *Ausführungspfade* gibt, mit denen sich genau die jeweilige Nachrichtenaustauschsequenz realisieren läßt.

Zunächst sollte das zu validierende System wieder in den Startzustand versetzt werden, da die *MSCs* ebenfalls vom Startzustand ausgehen. Dies geschieht durch den Menüpunkt *Restart* im Menü *File*. Die folgende Nachfrage wird bestätigt.

Der Button *Verify MSC* in der Abteilung *Explore* startet die Überprüfung eines *MSCs*. In der Dateiauswahl soll zunächst der

Tools→*Editors*→*MSC Editor*

File→*Open*

Doppelklick auf *TraceLeadingToSuccess.msc*

bei gedrückter *Shift*-Taste Pfeile der Nachrichten *Getraenkanforderung* und *Getraenk* selektieren

rechte Maustaste→*Copy*

rechte Maustaste→*Paste*

unter *warten* plazieren

File→*Save as...*, *TraceLeadingToFail.msc*

File→*Exit*

Untitled-Zeile selektieren

Edit→*Remove...*

Remove

File→*Restart*

OK



Verify MSC

MSC TraceLeadingToSuccess.msc ausgewählt werden.

Der Validator überprüft nun die Vereinbarkeit des *MSCs* mit dem *SDL-System* und sollte Erfolg melden.

Bevor die Vereinbarkeit mit dem zweiten *MSC* überprüft wird, muß das zu validierende System erneut in den Startzustand versetzt werden. Dies geschieht abermals durch den Menüpunkt *Restart* im Menü *File*. Die folgende Nachfrage wird ebenfalls bestätigt.

Nun kann geprüft werden, ob der zweite *MSC TraceLeadingToFail.msc* mit dem *SDL-System* vereinbar ist. Wieder den Button *Verify MSC* in der Abteilung *Explore* drücken. In der Dateiauswahl nun *TraceLeadingToFail.msc* auswählen.

Der Validator meldet, daß kein Pfad, der mit dem *SDL-System* vereinbar ist, gefunden werden konnte – was wir ja auch erwartet haben. Im *Report Viewer* wird außerdem einen *Deadlock* angezeigt. Ein Doppelklick auf den zugehörigen Kasten klappt diesen aus. Ein weiterer Doppelklick auf den gerade herausgeklappten Kasten zeigt die Stelle im *MSC* bis zu der der Validator bisher gelaufen ist. Da es von dieser Stelle aus keine Möglichkeit gibt, das geforderte *Signal Getraenk* zu erhalten, wird ein *Deadlock* gemeldet.

7 Beenden

Der *Organizer* kann per *Exit* im Menü *File* beendet werden. Evtl. Nachfragen zum Speichern können bestätigt werden, falls es noch ungespeicherte Änderungen gibt.

Doppelklick auf *TraceLeadingToSuccess.msc*

```
** MSC ValidatorTrace verified **
```

File→*Restart*
OK

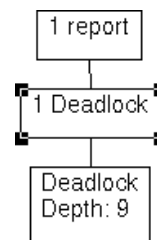


Verify MSC

Doppelklick auf *TraceLeadingToFail.msc*

```
** MSC ValidatorTrace NOT VERIFIED **
```

Doppelklick auf *1 Deadlock*



noch ein Doppelklick

File→*Exit*