Übungen zur Vorlesung
Testautomatisierung
Wintersemester 2002/03

AG BS
Jan Peleska, Aliki Tsiolakis

Blatt 1

CSP-Spezifikationen zum Testen eines *Engine Controller*

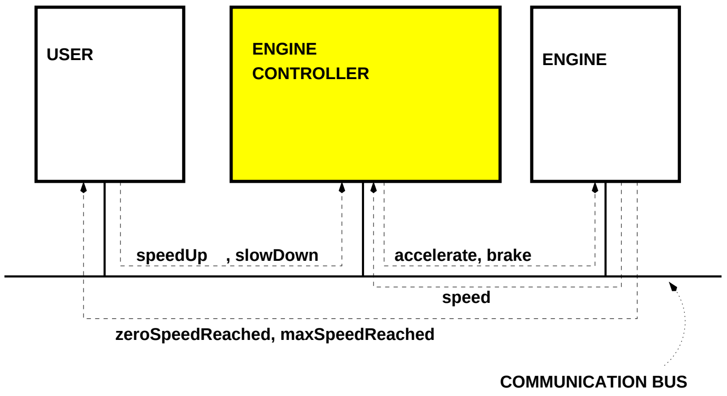
Die Firma *TA Mobil* hat ein neues Auto auf den Markt gebracht, das besonders auf Sonntagsfahrer im Stadtverkehr zugeschnitten ist: Die Maximalgeschwindigkeit von 50km/h darf nicht länger als 5sec gefahren werden.
Das Auto kann durch drei spezielle Kontroll-Kommandos vom Fahrer „gesteuert“ werden:

- Mit dem Kontroll-Kommando speedUp (x) kann das Auto so beschleunigt werden, dass es x km/h schneller fährt als vorher.
- Umgekehrt kann die Geschwindigkeit des Autos um x km/h gedrosselt werden, indem das Kontroll-Kommando slowDown (x) an den Motor weitergeleitet wird. (Allerdings soll der *Engine Controller* keine Signale an den Motor weiterleiten, wenn das Auto bereits steht.)
- Mit dem Kontroll-Kommando steady wird die aktuelle Geschwindigkeit beibehalten.

Diese Kontroll-Kommandos werden an den *Engine Controller* gesendet, der daraufhin den Motor beschleunigt (accelerate) oder abbremst (brake). Die Umsetzung der Kontroll-Kommandos muss innerhalb von $t_1 = 1sec$ erfolgen.
Sobald das Auto die maximale oder minimale Geschwindigkeit erreicht, gibt der Motor ein kurzes Signal, das je nach persönlicher Einstellung des Fahrers durch ein kurzes akkustisches oder visuelles Signal angezeigt werden könnte:

- Das Signal maxSpeedReached zeigt an, dass die maximale Geschwindigkeit von 50km/h erreicht wurde.
- Das Signal zeroSpeedReached zeigt an, dass die minimale Geschwindigkeit (0km/h) erreicht wurde und das Auto steht.

Als erzieherische Maßnahme für Raser (und zur Schonung des Motors) kann die maximale Geschwindigkeit aber nicht länger 5sec gefahren werden. Deshalb muss der Fahrer innerhalb von 5sec nach Erreichen der maximalen Geschwindigkeit (also innerhalb von 5 sec nach dem Signal maxSpeedReached), wieder eine Abbremsung des Autos auslösen (Kontroll-Kommando slowDown). Falls dies nicht geschieht, leitet der *Engine Controller* eine Notbremsung ein, die dazu führt, dass das Fahrzeug mit maximaler Verzögerung abgebremst wird.
Für die Darstellung der aktuellen Geschwindigkeit x auf dem Tacho sendet der Motor jede Sekunde ($t_0 = 1sec$) das Signal speed (x) an den *Engine Controller*.
Zum Testen der Engine Controller-Software steht die Original-Hardware des Motors leider nicht zur Verfügung. Das Verhalten des Motors wird in dem CSP-Prozess `ENGINE` simuliert (siehe Datei `engine.csp`). Das CSP-Interface für die *Engine Controller*-Software steht in der Datei `ifm_sut.csp` zur Verfügung.



Aufgabe 1: CSP-Spezifikation des Fahrers (`user.csp`) 50%

Teilaufgabe 1:

Beschreiben Sie die relevanten Fälle des Fahrerverhaltens, die auf der obigen Beschreibung des Systems basieren.

Teilaufgabe 2:

Schreiben Sie eine CSP-Spezifikation des Fahrers (Prozess `USER`), die das relevante Fahrerverhalten spezifiziert. Achten Sie darauf, dass das Interface dieser Spezifikation zu den Interfaces in `engine.csp` und `ifm_sut.csp` passt.
Dokumentieren Sie die CSP-Spezifikation sowie die dazugehörigen Datentyp- und Kanal-Deklarationen.

Aufgabe 2: CSP-Spezifikation des *Test Oracle* (`checker.csp`) 50%

Teilaufgabe 1:

Beschreiben Sie die Korrektheits-Bedingungen des*Engine Controller*, die durch das *Test Oracle* automatisch überprüft werden sollen. Halten Sie sich dabei an die obige Beschreibung des Systems.

Teilaufgabe 2:

Schreiben Sie eine CSP-Spezifikation des *Checkers* (Prozess `CHECKER`), die die Korrektheitsbedingungen des *Engine Controller* überprüfen kann.
Dokumentieren Sie die CSP-Spezifikation sowie die dazugehörigen Datentyp- und Kanal-Deklarationen.

Abgabe: Bis Dienstag, 12. November 2002, vor dem Tutorium.

Geben Sie für alle Aufgaben eine **schriftliche Lösung** ab. Bitte schicken Sie zusätlich die csp-Dateien sowie das PDF oder Postscript ihrer Lösung an `tsio@informatik.uni-bremen.de`.

Sowohl bei der schriftlichen Lösung als auch in den CSP-Dateien die Namen aller Gruppenmitglieder nicht vergessen!

engine.csp

```
-----------------------------------------------------------------------
--
-- SIMULATION OF ENGINE BEHAVIOUR
--
-- Testautomatisierung, WS 2002/03, Jan Peleska und Aliki Tsiolakis
-- copyright: Verified Systems International GmbH
--            Universität Bremen, TZI
--
-----------------------------------------------------------------------

-----------------------------------------------------------------------
-- TYPE DEFINITIONS
-----------------------------------------------------------------------

-- timer definitions

t0 = 0                -- Duration of timer 0 is 1sec
TIMER = { 0..9 }

-----------------------------------------------------------------------
-- CHANNEL DECLARATIONS
-----------------------------------------------------------------------

--
-- timer
--
pragma AM_SET_TIMER
channel setTimer : TIMER

pragma AM_ELAPSED_TIMER
channel elapsedTimer : TIMER

--
-- Output of this Abstract Machine
--
pragma AM_OUTPUT
channel maxSpeedReached
channel zeroSpeedReached
channel speed : { 0..50 }

--
-- Input to this Abstract Machine
--
pragma AM_INPUT
channel accelerate : {0..5}
channel brake      : {0..5}

--
-- Internal Events
--
pragma AM_INTERNAL
channel thisSpeed : {0..50}

-----------------------------------------------------------------------

ENGINE = ((zeroSpeedReached -> STOPPED)
           [|{| thisSpeed, elapsedTimer |}|]
          SEND_SPEED) \ {| thisSpeed |}


-----------------------------------------------------------------------

STOPPED =
        accelerate?x -> ( if ( 0 < x )
                          then SPEEDING(x,0)
                          else STOPPED )
[]
elapsedTimer.t0 -> STOPPED

-----------------------------------------------------------------------

SPEEDING(a,s) =
(a+s < 50) & elapsedTimer.t0
                        -> thisSpeed!s+a
                        -> SPEEDING(a,s+a)
        []
(a+s >= 50) & elapsedTimer.t0
                        -> maxSpeedReached
                        -> thisSpeed!50
                        -> MAXSPEED
[]
accelerate?x -> SPEEDING(x,s)
[]
brake?x -> BRAKING(x,s)

-----------------------------------------------------------------------

MAXSPEED =
  elapsedTimer.t0 -> MAXSPEED
[]
accelerate?x -> MAXSPEED
[]
brake?x -> ( if ( 0 < x )
                     then BRAKING(x,50)
     else MAXSPEED )

-----------------------------------------------------------------------

BRAKING(a,s) =
(s-a > 0) & elapsedTimer.t0
                        -> thisSpeed!s-a
                        -> BRAKING(a,s-a)
[]
(s-a <= 0) & elapsedTimer.t0
                        -> zeroSpeedReached
                        -> thisSpeed!0
                        -> STOPPED
[]
accelerate?x -> SPEEDING(x,s)
[]
brake?x -> BRAKING(x,s)

-----------------------------------------------------------------------
```

```
SEND_SPEED  = setTimer!t0 -> speed!0 -> SP(0)

SP(x) =
elapsedTimer.t0 -> setTimer!t0 -> speed!x -> SP(x)
[]
thisSpeed?y -> SP(y)

----------------------------------------------------------------------
```

ifm_sut.csp

```
----------------------------------------------------------------------
--
-- ABSTRACT INTERFACE TO SUT
--
-- Testautomatisierung, WS 2002/03, Jan Peleska und Aliki Tsiolakis
-- copyright: Verified Systems International GmbH
--            Universität Bremen, TZI
--
----------------------------------------------------------------------

----------------------------------------------------------------------
-- TYPE DEFINITIONS
----------------------------------------------------------------------

datatype accel = speedUp |  slowDown | steady


----------------------------------------------------------------------
-- CHANNEL DECLARATIONS
----------------------------------------------------------------------

--
-- Input for the SUT
--

-- from USER
channel ctrl : accel.{0..5}

-- from engine
channel maxSpeedReached
channel zeroSpeedReached
channel speed : { 0..50 }


--
-- Output of the SUT
--

-- to engine
channel accelerate : {0..5}
channel brake      : {0..5}

----------------------------------------------------------------------
```