Tentamen Automated Reasoning, 4 februari 2004, 14 - 17 uur

1. We beschouwen een PVS-theorie met typeparameter `T` en de volgende declaraties:

```
P: TYPE = [nat -> T]
m, n: VAR nat
t: VAR T
pi: VAR P
rr: VAR pred[nat,P]
```

a) Formuleer een PVS-lemma met de inhoud:

$$\forall n\ rr(n, \pi(n)) \rightarrow \forall n\ \exists t\ rr(n, t)$$

Schets hoe dit lemma met PVS bewezen kan worden. Geef aan hoe de bewijsboom eruit gaat zien, wat de optredende sequenten zijn, en welke commando's je gebruikt om vanuit een sequent verder te komen. Je mag standaardsymbolen (als $\forall$) gebruiken voor PVS-keywords als `FORALL`, etc. Je hoeft de typering niet mee te slepen in de sequenten.

b) We bekijken nu de semantiekdefinitie van CTL*:

$t \models \neg\varphi$	iff	not $t \models \varphi$
$t \models \varphi_1 \wedge \varphi_2$	iff	$t \models \varphi_1$ and $t \models \varphi_2$
$t \models E\psi$	iff	there is a path π from t such that $\pi \models \psi$
$t \models A\psi$	iff	for every path π from t, $\pi \models \psi$

$\pi \models \neg\psi$	iff	not $\pi \models \psi$
$\pi \models \psi_1 \wedge \psi_2$	iff	$\pi \models \psi_1$ and $\pi \models \psi_2$
$\pi \models X\psi$	iff	$\pi^1 \models \psi$
$\pi \models F\psi$	iff	for some $n \in \mathbb{N}, \pi^n \models \psi$
$\pi \models G\psi$	iff	for all $n \in \mathbb{N}, \pi^n \models \psi$
$\pi \models \psi_1 U \psi_2$	iff	for some $n \in \mathbb{N}, \pi^n \models \psi_2$ and for all m such that $0 \leq m < n$, $\pi^m \models \psi_1$

We gaan deze semantiek formaliseren in PVS. Daartoe onderscheiden we *toestands*formules `tform` (hierboven φ) en *pad*formules `pform` (hierboven ψ):

```
tform: VAR pred[T]
pform: VAR pred[P]
```

Dus toestandsformules worden predikaten op (ofwel verzamelingen van) toestanden, en padformules worden predikaten op paden. Voor elke logische operator uit de semantiekdefinitie introduceren we een operator in PVS:

```
niet(tform): pred[T]
en(tform!1,tform!2): pred[T]
E(pform): pred[T]
A(pform): pred[T]

niet(pform): pred[P]
en(pform!1,pform!2): pred[P]
X(pform): pred[P]
F(pform): pred[P]
G(pform): pred[P]
U(pform!1,pform!2): pred[P]
```

Geef nu de definities van deze operatoren. Als voorbeeld hier de eerste:

```
niet(tform): pred[T] = {t | NOT pform(t)}
```

Definieer en gebruik daarbij een hulpfunctie voor de n-de suffix π^n van een pad π.
c) Geef een definitie van het predikaat

```
fair(tform): pred[P]
```

als de verzameling van paden waarvoor geldt dat `tform` er oneindig vaak geldt.

2. Dijkstra publiceerde in 1974 het eerste zelf-stabiliserende algoritme, dat als volgt beschreven kan worden. Gegeven zijn natuurlijke getallen $N < K$. Er is een ongeïnitialiseerd array `a[0..N]` van integers. Het algoritme is nondeterministisch met stappen die voldoen aan

$$\begin{array}{l}\textbf{do}\\ [\!] \quad i : i < N \quad \wedge \quad \mathtt{a}[i] \neq \mathtt{a}[i+1] \quad \rightarrow \quad \mathtt{a}[i] := \mathtt{a}[i+1]\\ [\!] \quad \mathtt{a}[N] = \mathtt{a}[0] \quad \rightarrow \quad \mathtt{a}[N] := (\mathtt{a}[0]+1) \ \textbf{mod} \ K\\ \textbf{od} \ .\end{array}$$

Het idee is dat $N+1$ processen in een ring staan en elk hun eigen array-element hebben, en dat telkens wijzigen op grond van de waarde van de rechter buur. De zelf-stabilisatie houdt in dat de toestand op den duur voldoet aan het predicaat

$$\text{(Stab)} \qquad (\exists\, j \ :: \ (\forall\, i : j \leq i : \mathtt{a}[i] = \mathtt{a}[N]) \ \wedge \ (\forall\, i : i < j : (\mathtt{a}[i]+1) \ \textbf{mod} \ K = \mathtt{a}[N])) \ ,$$

waarbij i en j over $[0\,..\,N]$ lopen. Als dit predicaat geldt, is het process met het nummer $(j-1)$ **mod** K als enige in staat om een stap te doen. Dit kan bv. voor wederzijdse uitsluiting gebruikt worden.

(a) Modelleer dit algoritme in Promela met behulp van $N+1$ processen die elk verantwoordelijk zijn voor de wijzigingen van één array-element. Zorg ten behoeve van de modellering dat elk process eerst zijn eigen array-element nondeterministisch initialiseert met een waarde $< K$. Geef aan hoe je met Spin test dat altijd tenminste één van de processen een stap kan doen.

(b) Geef temporele formules die uitdrukken:
(1) als de toestand eenmaal aan (Stab) voldoet, blijft dat zo.
(2) elke berekening geraakt op den duur in een toestand die aan (Stab) voldoet.
Gebruik hiertoe temporele formules zoals Spin die accepteert.

(c) Geef aan hoe de beweringen (1) en (2) met Spin getest kunnen worden. Je kunt desgewenst variabelen aan je Promelaprogramma toevoegen. Berekeningen dienen niet onnodig inefficiënt te zijn.