Aufgabe 14: Sattelpunkt einer Matrix

Sei $\mathcal{A}[0\ldots N-1, 0\ldots N-1]$ eine Matrix, deren Einrtäge Integer-Werte sind. Ein Element $A[u,v]$ nennt man einen Sattelpunkt der Matrix, genau dann wenn gilt:

$$\begin{aligned} A[u,v] &= \langle min_i : 0 \leq i < N :: A[i,v]\rangle \wedge \\ A[u,v] &= \langle max_j : 0 \leq j < N :: A[u,j]\rangle \end{aligned}$$

Geben Sie ein Programm an, das bestimmt, ob eine gegebene Matrix einen Sattelpunkt besitzt. D.h. berechnen Sie den Wert der Boolean Variablen sp, gegeben durch:

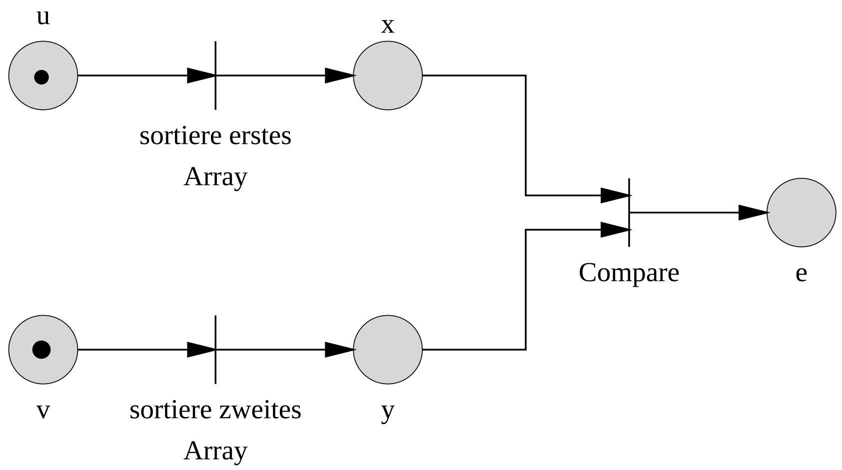
$$\begin{aligned} sp \;=\; & \langle \exists u,v : 0 \leq u < N \wedge 0 \leq v < N :: \\ & A[u,v] = \langle min_i : 0 \leq i < N :: A[i,v]\rangle \wedge \\ & A[u,v] = \langle max_j : 0 \leq j < N :: A[u,j]\rangle\rangle \end{aligned}$$

Aufgabe 15: Erzeugung von Primzahlen:

In dieser Aufgabe soll ein Array $X[1\ldots N]$ erzeugt werden, das die ersten N Primzahlen enthält ($N \geq 1$).

Aufgabe 16: Simulation eines Petrinetzes

Petrinetze sind ein eleganter Mechanismus zur Untersuchung von Synchronisationsmechanismen. Zur Darstellung soll ein konkretes Beispiel verwendet werden: Task Sequenzing in einem Programm.
Dazu ist es notwendig, zwei Arrays zu vergleichen, um festzustellen, ob sie dieselbe Menge von Elementen enthalten. Die Programmstrategie ist daher, jedes Array in aufsteigender Reihenfolge zu ordnen. Zum Vergleich der beiden geordneten Arrays verwenden wir dann ein bereits gegebenes Programm *Compare*. Das zu betrachtende Petrinetz ist im folgenden Bild gegeben:



Die Kreise repräsentieren Stellen und die Rechtecke Transitionen. Die Stellen u, v enthalten initial ein Token, die anderen Stellen sind initial leer. Eine Transition kann nur dann feuern, wenn alle Input-Stellen mindestens ein Token enthalten. Dabei wird ein Token von jeder Input-Stelle entfernt und ein Token zu jeder Output-Stelle hinzugefügt. Jede Transition, die diese Bedingung erfüllt, kann feuern. Aufgabe ist es nun, das dargestellte Petrinetz als UNITY Programm darzustellen.

Lösungsvorschlag Aufgabe 14:

Alle Indices i, j, u, v werden über einem Definitionsbereich $0 \ldots N-1$ definiert. Für jedes u, v gilt dabei:

$$\langle min_i :: A[i,v]\rangle \leq A[u,v] \leq \langle max_j :: A[u,j]\rangle$$

$A[u,v]$ ist daher genau dann ein Sattelpunkt, wenn gilt:

$$\langle min_i :: A[i,v]\rangle \geq \langle max_j :: A[u,j]\rangle$$

Wir erhalten daher:

$$sp = \langle \exists u, v :: \langle min_i :: A[i,v]\rangle \geq \langle max_j :: A[u,j]\rangle\rangle$$

Sei $X[v]$ gegeben als $\langle min_i :: A[i,v]\rangle$ und $Y[u]$ als $\langle max_j :: A[u,j]\rangle$. Dann ist

$$\begin{aligned} sp &= \langle \exists u, v :: X[v] \geq Y[u]\rangle \\ &= (\langle max_v :: X[v]\rangle \geq \langle min_u :: Y[u]\rangle) \end{aligned}$$

Zur Berechnung von sp kann damit folgendes Programm angegeben werden:

Program Sattelpunkt
declare X,Y : array[0...N-1] of integer ;
always
$\langle \| v :: X[v] = \langle min_i :: A[i,v]\rangle\rangle$
$\| \langle \| u :: Y[u] = \langle max_j :: A[u,j]\rangle\rangle$
$[] sp = (\langle max_v :: X[v]\rangle \geq \langle min_u :: Y[u]\rangle)$
end{Sattelpunkt}

Auf einer Monoprozessormaschinee kann das Programm mit einem Aufwand $O(N^2)$ ausgeführt werden, auf einer Parallelmaschine mit $O(N^2)$ Prozessoren mit $O(\log N)$

Lösungsvorschlag Aufgabe 15:

Zunächst gilt: $X[1] = 2$. Die $(n+1)^{te}$ Primzahl, $n \geq 1$, kann nun als eine Menge von Gleichungen gegeben werden, wobei u div v angibt, daß u v teilt.

$$\begin{aligned} X[1] &= 2 \\ & [] \langle [] n : 1 \leq n < N :: X[n+1] = \\ & \langle min_p : p > X[n] \wedge \langle \forall k : 1 \leq k \leq n :: \neg X[k] \text{ div } p\rangle :: p\rangle\rangle \end{aligned}$$

Diese Gleichungen können nicht direkt als UNITY Programm verwendet werden, da eine unendliche Anzahl von p die Bedingung in der Quantifizierung erfüllt. Wir implementieren diese Gleichungen als ein Programm mit Zuordnungen. Dazu führen wir die Variablen n, p, i ein, wobei n die Anzahl der bisher berechneten Primzahlen darstellt, p bezeichnet einen Kandidaten für die nächste, noch nicht berechnete Primzahl. Formal haben wir dann die folgende Invariante:

$$\begin{array}{rl} p > X[n] \wedge 1 < i \leq n+1 \wedge 1 \leq n \leq N & \wedge \\ \langle \wedge k : 1 \leq k \leq n :: X[k] \text{ ist die } k^{te} \text{ Primzahl} & \wedge \\ \langle \wedge q : X[n] < q < p :: q \text{ ist nicht prim } \rangle & \wedge \\ \langle \wedge k : 1 \leq k < i :: \neg X[k] \text{ div } p \rangle & \end{array}$$

Beachte, daß aus der Invarianten $\neg X[1]$ div p folgt und daher ist p ungerade.

Program Primzahlen
  declare n,p,i : integer ;
  initially n,p,i = 1,3,2 $\| X[1] = 2$
  assign

$$\begin{array}{rcrcrcrcll} X[i] & , & i & , & p & , & n & := & & \\ X[i] & , & 2 & , & p+2 & , & n & \text{if} & i \leq n \wedge X[i] \text{ div } p & \sim \\ X[i] & , & i+1 & , & p & , & n & \text{if} & i \leq n \wedge \neg X[i] \text{ div } p & \sim \\ p & , & 2 & , & p+2 & , & n+1 & \text{if} & i > n \wedge n < N & \end{array}$$

end{Primzahlen}

Lösungsvorschlag Aufgabe 16:

Jede Stelle wird im Programm repräsentiert als eine Variable, die einen Integer Wert annehmen kann. Dieser Wert gibt die Anzahl der Token in einer Stelle an. Eine Transition wird modelliert durch einen Ausdruck, der jede Variable um 1 reduziert, entsprechend den Input-Stellen und um 1 erhöht, entsprechend den Output-Stellen.

Program Petrinetz
  initially u,v,x,y,e = 1,1,0,0,0 ;
  assign

$$\begin{array}{rcrlrr} u, x & := & u-1, x+1 & \text{if} & u > 0 & \{\text{sortiere erstes Array}\} \\ [] v, y & := & v-1, y+1 & \text{if} & v > 0 & \{\text{sortiere zweites Array}\} \\ [] x, y, e & := & x-1, y-1, e+1 & \text{if} & x > 0 \wedge y > 0 & \{\text{Compare}\} \end{array}$$

end{Petrinetz}