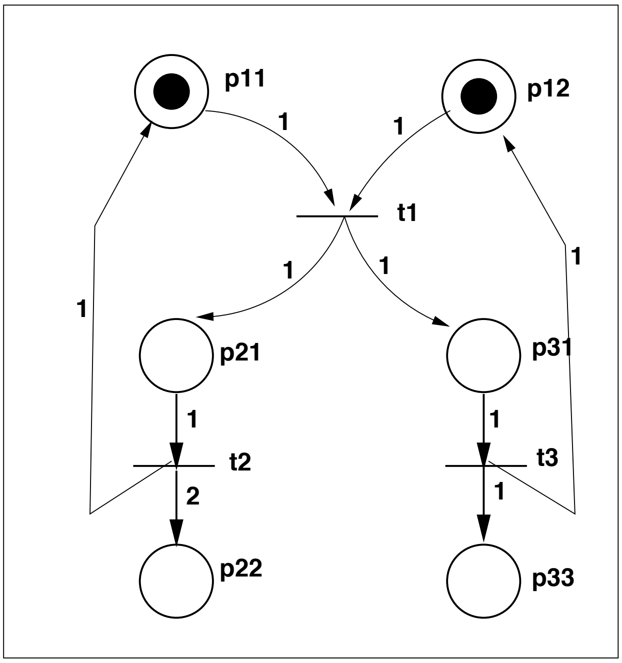

.

Cours Modélisation et vérification des systèmes informatiques
Exercices
Modélisation TLA$^+$ (2)
par Dominique Méry
22 septembre 2021

Exercice 1 *Soit le réseau de Petri de la figure 1.*

Question 1.1 *Déterminer les conditions initiales.*

Question 1.2 *Déterminer les relations modélisant les transitions.*

Question 1.3 *Valider les propriétés et les hypothèses que vous pourrez faire sur ce réseau de Petri.*

Consider the flowchart program (Fig. 3-18) over the integers for computing *McCarthy's 91 function*. Prove that it is totally correct wrt $\varphi(x): T$ and $\psi(x, z): z = \textit{if } x > 100 \textit{ then } x - 10 \textit{ else } 91$.

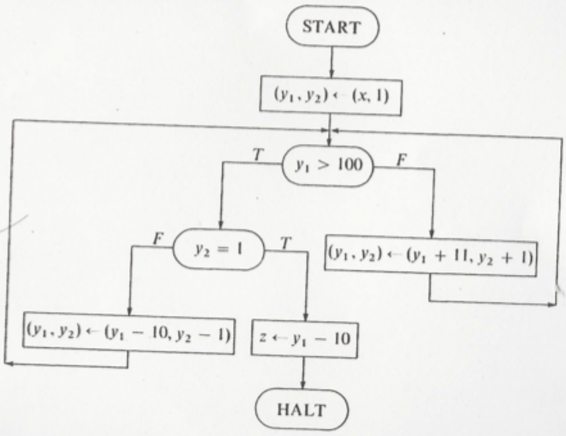

Figure 3-18 Flowchart program for computing McCarthy's 91 function.

Exercice 2

Question 2.1 *Construire un module TLA$^+$ modélisant les différents pas de calcul.*

Question 2.2 *Evaluer l'algorithme en posant des questions de sûreté suivantes :*

1. *l'algorithme est partiellement correct.*
2. *l'algorithme n'a pas d'erreurs à l'exécution.*

Exercice 3 *Soit le schéma suivant définissant un calcul déterminant sir un nombre entier naturel est premier ou non.*

Consider the flowchart program (Fig. 3-19) over the integers for deciding whether or not a given integer x is a prime number. Prove that it is totally correct wrt $\varphi(x): x \geq 2$ and $\psi(x, z): z \equiv prime(x)$, where $prime(x)$ is *true* if and only if x is a prime number; that is $\forall k[2 \leq k < x \supset rem(x, k) \neq 0]$.

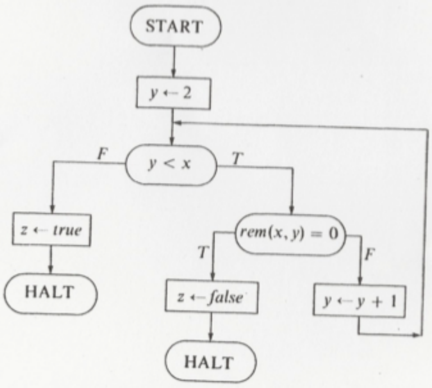

Figure 3-19 Flowchart program for deciding whether or not x is a prime number.

Question 3.1 *Ecrire un modèle TLA modélisant ce schéma de calcul.*

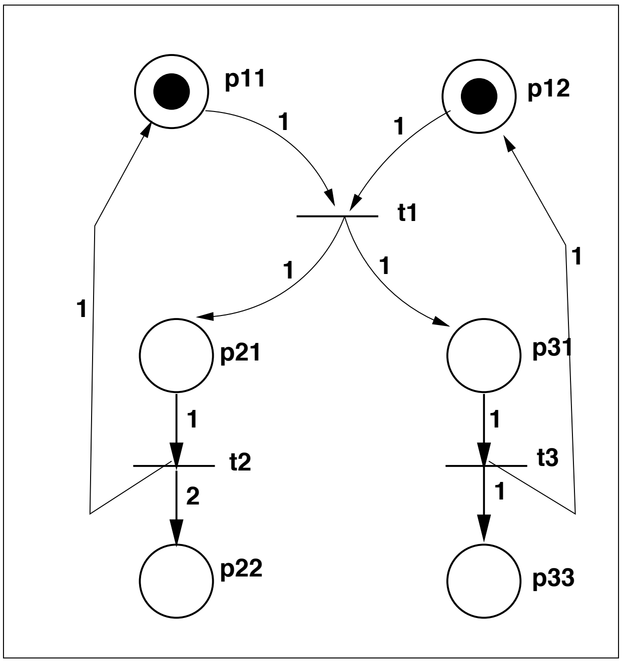

FIGURE 1 – Réseau de Petri

Question 3.2 *Ecrire un invariant à partir d'annotations que vous définirez après avoir défini des points de contrôle.*

Question 3.3 *Vérifier la correction partielle*

Question 3.4 *Vérifier l'absence d'erreurs à l'exécution.*

Exercice 4 *Le module truc permet de résoudre un problème très classique en informatique : trouver un chemin entre un sommet input et des sommets output supposés être des sommets de sortie.*

Question 4.1 *Pour trouver un chemin de input à l'un des sommets de output, il faut poser une question de sûreté à notre système de vérification. Donner une question de sûreté à poser permettant de trouver un chemin de input vers un sommet de output.*

Question 4.2 *On désire utiliser cette technique pour trouver un chemin dans un labyrinthe. Un labyrinthe est représenté par une matrice carrée de taille n. On définit ensuite pour chaque élément* $<< i, j >>$ *de la matrice les voisins communiquant à l'aide de la fonction lab qui associe à* $<< i, j >>$ *les éléments qui peuvent être atteints en un coup. Par exemple, le mouvement possible à partir de* $<< 1, 1 >>$ *est* $<< 2, 1 >>$*, ou le mouvement possible à partir de* $<< 2, 1 >>$ *est* $<< 2, 2 >>$ *ou* $<< 1, 1 >>$*, ou le mouvement possible à partir de* $<< 2, 2 >>$ *est* $<< 2, 3 >>$ *ou* $<< 3, 2 >>$ *ou* $<< 2, 1 >>$*, ...*

```
lab == [<<x,y>> \in (nodes \X nodes) |->
                    IF x=1 /\ y=1  THEN {<<2,1>>} ELSE
                    IF x=2 /\ y=1  THEN {<<2,2>>}
                    IF x=1 /\ y=2  THEN {} ELSE
                    IF x=2 /\ y=2  THEN {<<3,2>>,<<2,3>>} ELSE
                    ELSE {}

                    ]
```

MODULE *Naturals*

A dummy module that defines the operators that are defined by the real Naturals module.

$$
\begin{array}{l}
Nat \triangleq \{\,\} \\
a+b \triangleq \{a,\ b\} \\
a-b \triangleq \{a,\ b\} \\
a \cdot b \triangleq \{a,\ b\} \\
a^{b} \triangleq \{a,\ b\} \\
a < b \triangleq a = b \\
a > b \triangleq a = b \\
a \leq b \triangleq a = b \\
a \geq b \triangleq a = b \\
a \,\%\, b \triangleq \{a,\ b\} \\
a \div b \triangleq \{a,\ b\} \\
a \mathrel{..} b \triangleq \{a,\ b\}
\end{array}
$$

MODULE *TLC*

LOCAL INSTANCE *Naturals*
LOCAL INSTANCE *Sequences*

$$
\begin{array}{l}
Print(out,\ val) \triangleq val \\
PrintT(out) \triangleq TRUE \\
Assert(val,\ out) \triangleq \text{IF } val = TRUE \text{ THEN } TRUE \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{ELSE CHOOSE } v : TRUE \\
JavaTime \triangleq \text{CHOOSE } n : n \in Nat \\
TLCGet(i) \triangleq \text{CHOOSE } n : TRUE \\
TLCSet(i,\ v) \triangleq TRUE
\end{array}
$$

$$
\begin{array}{l}
d :> e \triangleq [x \in \{d\} \mapsto e] \\
f \mathbin{@@} g \triangleq [x \in (\text{DOMAIN } f) \cup (\text{DOMAIN } g) \mapsto \\
\qquad\qquad\qquad \text{IF } x \in \text{DOMAIN } f \text{ THEN } f[x] \text{ ELSE } g[x]] \\
Permutations(S) \triangleq \\
\quad \{f \in [S \to S] : \forall\, w \in S : \exists\, v \in S : f[v] = w\}
\end{array}
$$

In the following definition, we use Op as the formal parameter rather than \prec because TLC Version 1 can't handle infix formal parameters.

$$
\begin{array}{l}
SortSeq(s,\ Op(\_,\ \_)) \triangleq \\
\quad \text{LET } Perm \triangleq \text{CHOOSE } p \in Permutations(1 \mathrel{..} Len(s)) : \\
\qquad\qquad\qquad\qquad \forall\, i,\ j \in 1..Len(s) : \\
\qquad\qquad\qquad\qquad\quad (i < j) \Rightarrow Op(s[p[i]],\ s[p[j]]) \lor (s[p[i]] = s[p[j]]) \\
\quad \text{IN } [i \in 1..Len(s) \mapsto s[Perm[i]]]
\end{array}
$$

$$RandomElement(s) \triangleq \text{CHOOSE } x \in s : TRUE$$

$$Any \triangleq \text{CHOOSE } x : TRUE$$

$$ToString(v) \triangleq (\text{CHOOSE } x \in [a : v,\ b : STRING] : TRUE).b$$

$$TLCEval(v) \triangleq v$$

FIGURE 2 – Modules Naturals.tla et TLC.tla

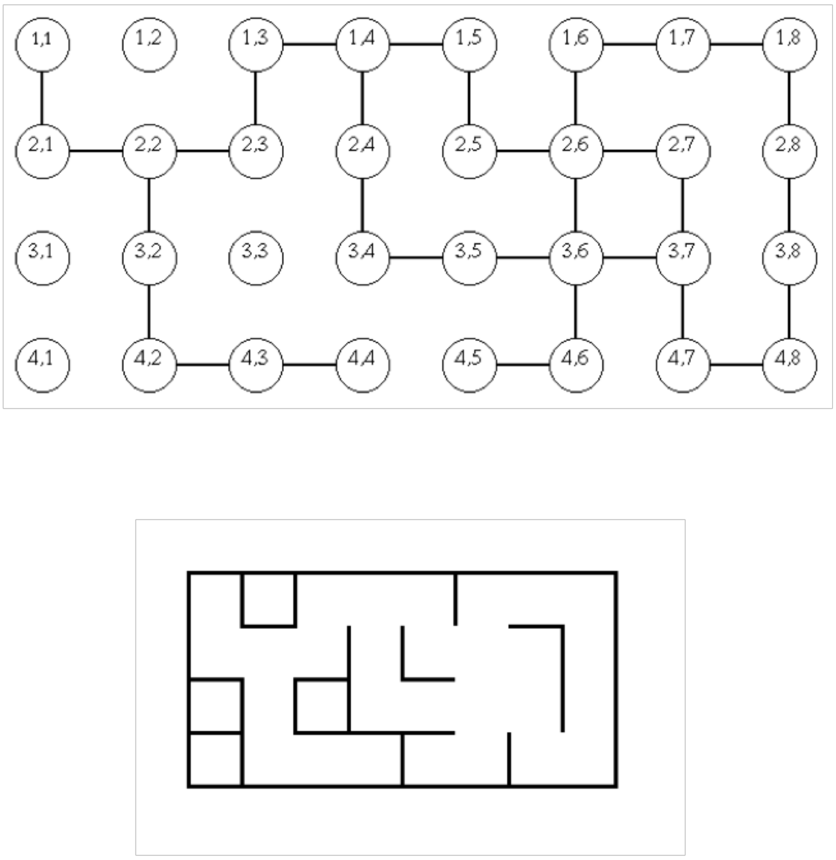

FIGURE 3 – Labyrinthe

Modifier le module truc pour traiter ce problème et donner la question à poser pour trouver une sortie.

MODULE *truc*

EXTENDS *Integers*, *TLC*
VARIABLES p
CONSTANTS *input*, *output*

$n \triangleq 10$
$nodes \triangleq 1..n$
$l \triangleq [i \in 1..n \mapsto$ IF $i = 1$ THEN $\{4, 5\}$ ELSE
IF $i = 2$ THEN $\{6, 7, 10\}$ ELSE
IF $i = 4$ THEN $\{7, 8\}$ ELSE
IF $i = 5$ THEN $\{\}$ ELSE
IF $i = 6$ THEN $\{4\}$ ELSE
IF $i = 7$ THEN $\{5\}$ ELSE
IF $i = 8$ THEN $\{5, 2\}$ ELSE
$\{\}$
$]$

$Init \triangleq p = 1$
$M(i) \triangleq \land\ i \in l[p]$
$\land\ p' = i$

$\textit{Next} \triangleq \exists\, i \in 1..n : M(i)$