Soutenance de Stage

Arthur Wenger & Karim Mamode

L3 Informatique Université de la Réunion

03 Juin 2017

Introduction

Objectifs

- Analyser un protocole de routage expérimental
- Modéliser un réseau
- Prouver des propriétés du réseau à travers la modélisation

Outils

- Le logiciel Coq pour l'élaboration de preuves
- Le langage de programmation fonctionnel Gallina

Analyse et Modélisation

Analyse du protocole

- Diviser le réseau en régions
- Aggréger les informations sur les noeuds d'une région
- Créer des routes vers des noeuds et des régions
- Appliquer un algorithme pour remplir les tables de routage

Modélisation

- Modéliser une région
- Réprésenter la structure des régions dans le réseau
- Décrire les liens entre les noeuds du réseau
- Représenter une route et un chemin entre deux noeuds

Les Régions

- Un type inductif pour le codage des régions

```
Inductive region : Type :=
| Z : region
| OO : region → region
| OI : region → region
| II : region → region
| IO : region → region.
```

- Un codage comparable à la base 4
- Permet de représenter l'imbrication des régions
- Un élément neutre pour le type inductif : Z

Interactions entre les régions

Objectifs

- Représenter la structure des régions dans le réseau
- Définir la notion de régions voisines
- Calculer les distances entre les régions

Moyens

- Listes de régions
- Matrices de régions
- Algorithme de partionnement du réseau

Listes

- Un type inductif polymorphe pour représenter des listes

```
Inductive clist (A:Type): Type :=
  | nil : clist A
  | cons : A → clist A → clist A.
```

- Interprétée comme un vecteur pour la construction de matrices
- Pas de contraintes sur le nombre d'élements

Listes de listes

- Un type inductif polymorphe pour représenter des listes de listes

```
Inductive listlist (A:Type) : Type :=
  | lnil : listlist A
  | lcons : clist A → listlist A → listlist A.
```

- Pas nécessairement homogènes
- Un ensemble englobant les matrices
- Pas de contraintes sur le nombres de listes

Matrices

- Un sous ensemble des listes de listes
- Nécessité de vérifier qu'une liste de listes est une matrice

```
Fixpoint is_matrix {A:Type}(m:listlist A) : bool :=
  match m with
  | lnil ⇒ true
  | lcons l m' ⇒
      match m' with
      | lnil ⇒ true
      | lcons l' m'' ⇒ Nat.eqb (list_count l)
                               (list_count l')
                        && is_matrix m'
      end
  end.
```

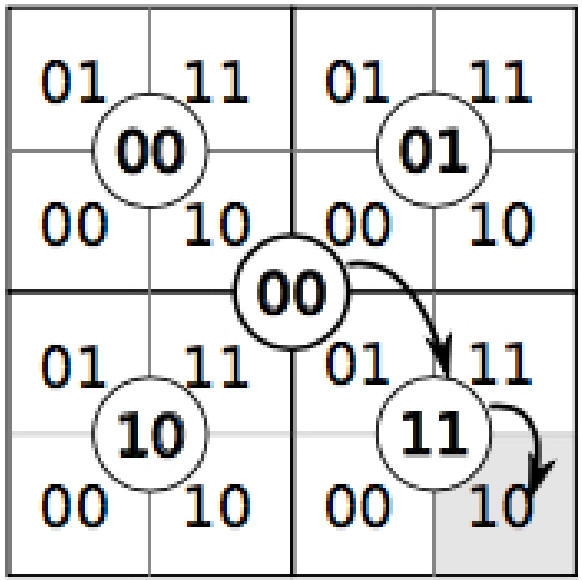
Matrices carrées

- Un sous ensemble des matrices
- Permet de représenter le partionnement du réseau

```
Definition is_square_matrix {A:Type}(m:listlist A):
bool :=
match m with
| lnil ⇒ true
| lcons l' m' ⇒ (is_matrix m) && (Nat.eqb (list_count l')
                                          (mat_count m))
end.
```

Partitionnement du réseau

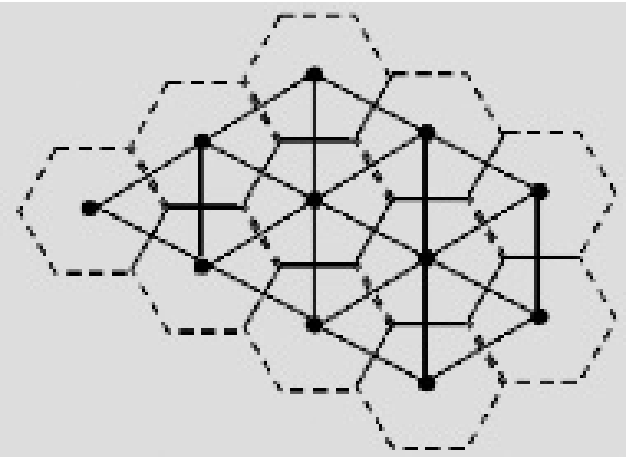
- Représenter l'organisation des régions sous forme matricielle



- Implémentation des partitionnements bottom-up et top-down
- Chaque région de la matrice finale a un numéro global
- Notion de distance entre 2 régions pour un niveau donné

Réseau

- On cherche à représenter le graphe réseau



- Représenter les liens entre les noeuds
- Modéliser un chemin entre 2 noeuds
- Définir les relations entre les noeuds et les régions

Noeuds et Chemins

- Un noeud est identifié par un entier
- Les liens sont modélisés par une fonction booléenne sur une paire de noeuds

```
Definition netnode := nat.
Definition graph:= netnode → netnode → bool.
```

- En ajoutant la propriété de transitivité on définit des chemins

```
Inductive has_path (g:graph) : nat → netnode → netnode →
Prop :=
| HP_Self (x:netnode): has_path g 0 x x
| HP_Step (n:nat)
          (x y z:netnode)
          (ST: g x y = true)
          (HP: has_path g n y z): has_path g (S n) x z.
```

Localisation

- Définir l'appartenance d'un noeud à une region

```
Definition loc_geo := region → netnode → bool.
```

- Modéliser les paramètres "k" et "m" du protocole
- Définir les contraintes

```
forall (x:netnode) (l:loc_geo), { r : region | l r x = true ∧
                                               rank r = m }.
```

- Deux noeuds sont-ils dans A0 ?

```
Definition is_in_A0 (r1 r2:region)(m: listlist region):
bool :=
match distance_regions_elem r1 r2 m with
| None ⇒ false
| Some d ⇒ d <? k
end.
```

Routes et Tables

- Un ensemble de destinations vers des noeuds ou des régions

```
Inductive table (l:loc_geo)(m: listlist region)(x:netnode):
Set :=
| rA0: ...  (* l'ensemble des noeuds du reseau
               situes dans A0 par rapport a x *)
| rAn: ...  (* l'ensemble des regions voisines
               de la region elementaire contenant x *)
```

- Le coût n'est pas représenté pour simplifier les premières preuves

Preuves

- Necessité d'itérer sur les étapes du cheminement d'un message
- Mesurer les variations de distances vers la destination
- Modéliser l'algorithme de contournement des vides
- Montrer qu'il ne peut pas y avoir de boucle réseau
- Montrer qu'il existe toujours des valeurs de k et de m qui permettent de trouver le chemin menant vers une destination