Une Analyse Formelle en Coq d'un Algorithme Distribué Probabiliste résolvant le Problème du Rendez-Vous

Allyx Fontaine
Akka Zemmari

JFLA 2013

03/02/2013

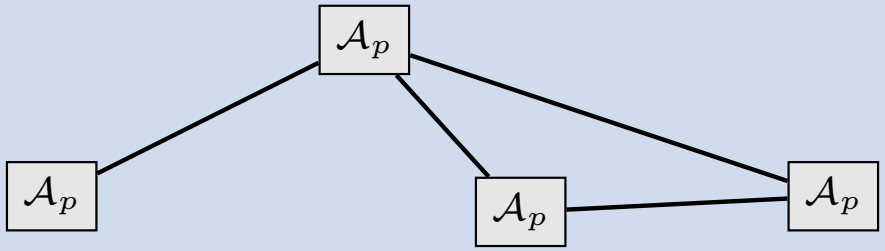
Plan

1. Contexte
2. Résultat principal
3. Modélisation en Coq
4. Preuves
5. Conclusion

Plan

1. Contexte
2. Résultat principal
3. Modélisation en Coq
4. Preuves
5. Conclusion

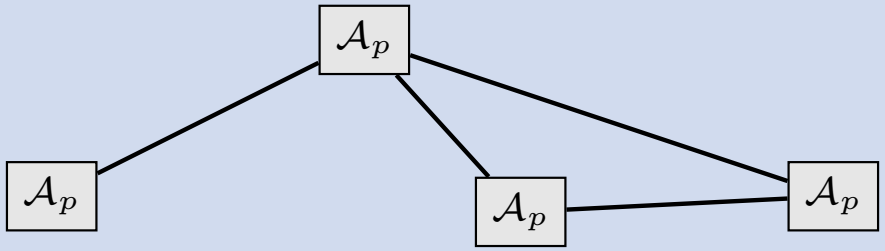
Algorithme distribué probabiliste



Hypothèses

- Réseau anonyme
- Pas de connaissance globale
- Communication par passage de messages
- Synchronisation par ronde

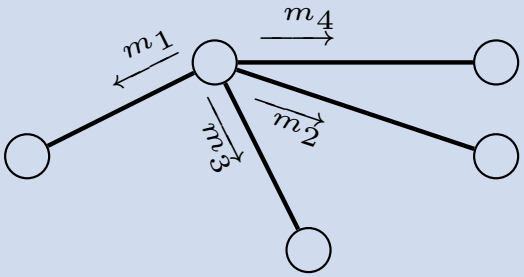
Algorithme distribué probabiliste



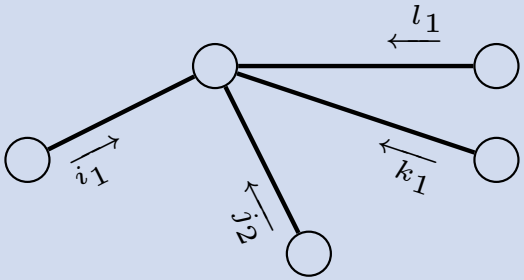
Hypothèses

- Réseau anonyme
- Pas de connaissance globale
- Communication par passage de messages
- Synchronisation par ronde

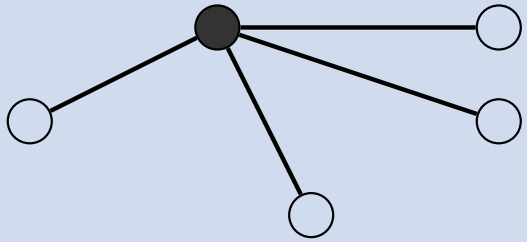
Une ronde pour un processeur



1. Envoi de messages



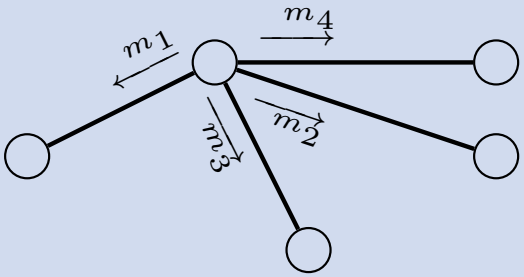
2. Réception de messages



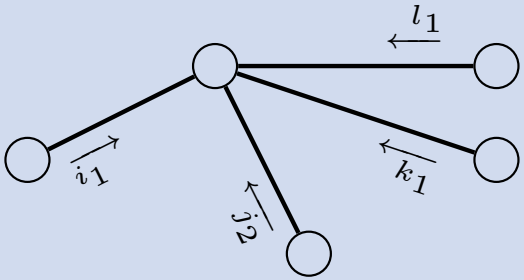
3. Calcul

Représentation non ambigüe d'une ronde

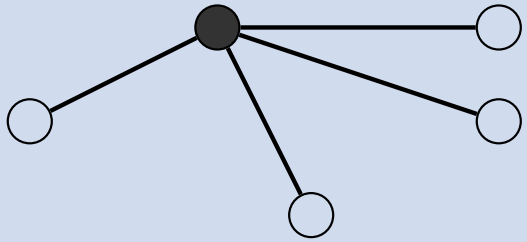
Une ronde pour un processeur



1. Envoi de messages

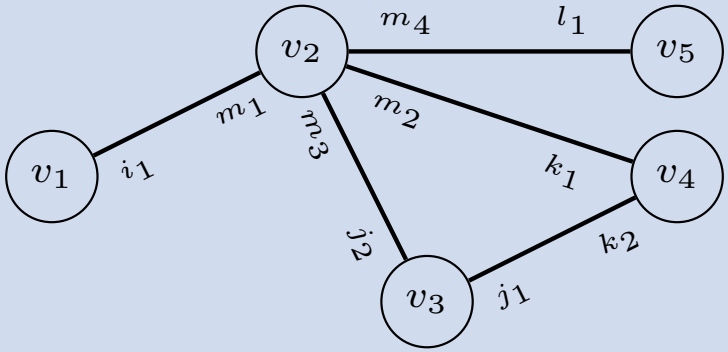


2. Réception de messages



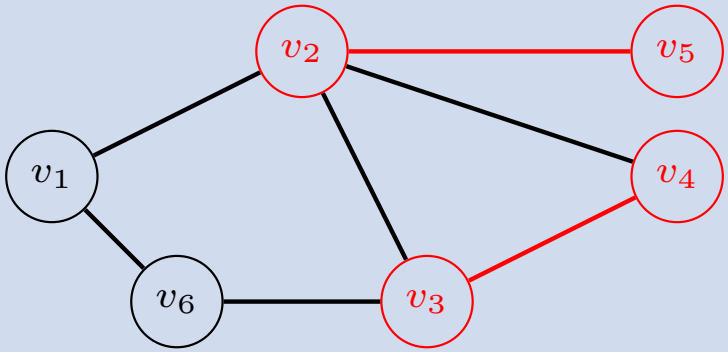
3. Calcul

Représentation non ambigüe d'une ronde



Définition

Un *rendez-vous* permet de réaliser des communications exclusives entre des paires de sommets voisins.



Intérêt

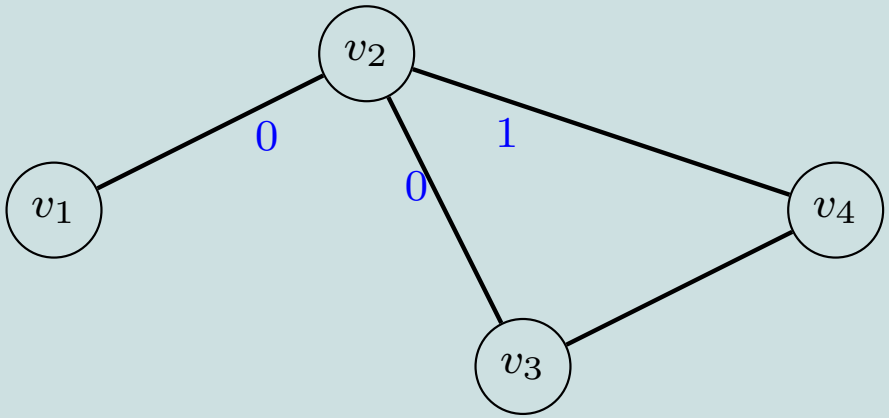
- Brique de base pour la mise en œuvre de systèmes de réécriture impliquant des règles sur les arêtes ([MS97])
- Réalisation (après un nombre d'itérations suffisant) d'un couplage maximal du graphe représentant le réseau

Algorithme

Chaque sommet v :

- choisit uniformément au hasard l'un de ses voisins, le $i$ème ;
- envoie 0 à tous ses voisins sauf au $i$ème à qui il envoie 1 ;

Exemple



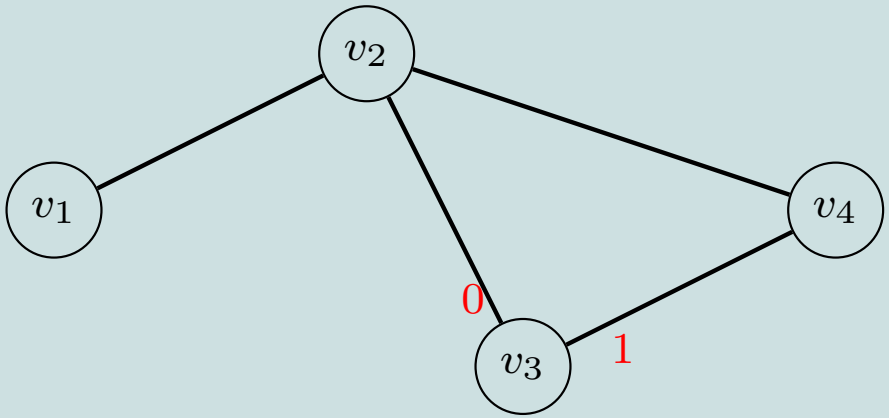
$c(v_1) = v_2$
$c(v_2) = v_4$
$c(v_3) = v_4$
$c(v_4) = v_3$

Algorithme

Chaque sommet v :

- choisit uniformément au hasard l'un de ses voisins, le $i$ème ;
- envoie 0 à tous ses voisins sauf au $i$ème à qui il envoie 1 ;

Exemple



$c(v_1) = v_2$
$c(v_2) = v_4$
$c(v_3) = v_4$
$c(v_4) = v_3$

Contexte

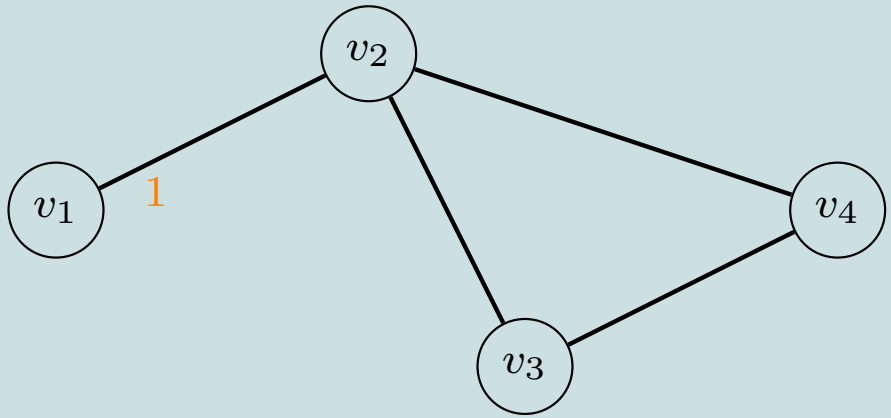
Un algorithme pour le problème du rendez-vous

Algorithme

Chaque sommet v :

- choisit uniformément au hasard l'un de ses voisins, le $i$ème ;
- envoie 0 à tous ses voisins sauf au $i$ème à qui il envoie 1 ;

Exemple



$c(v_1) = v_2$
$c(v_2) = v_4$
$c(v_3) = v_4$
$c(v_4) = v_3$

Contexte

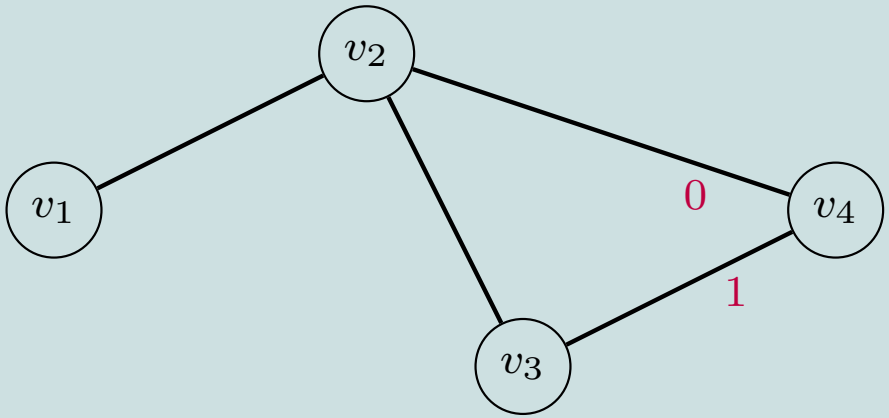
Un algorithme pour le problème du rendez-vous

Algorithme

Chaque sommet v :

- choisit uniformément au hasard l'un de ses voisins, le $i$ème ;
- envoie 0 à tous ses voisins sauf au $i$ème à qui il envoie 1 ;

Exemple



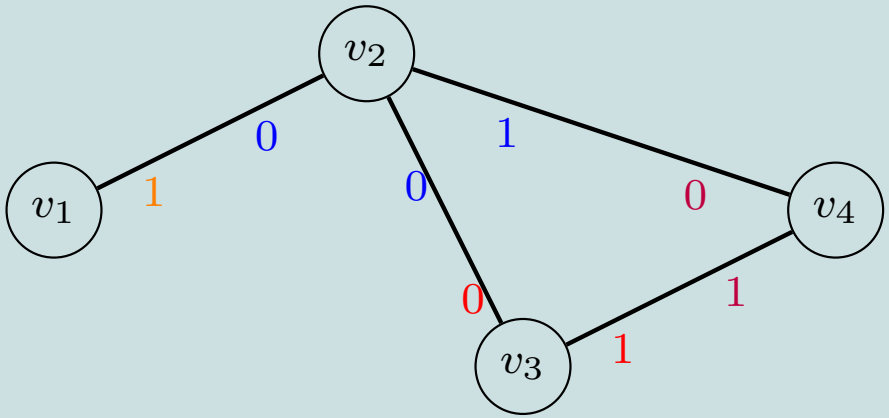
$c(v_1) = v_2$
$c(v_2) = v_4$
$c(v_3) = v_4$
$c(v_4) = v_3$

Algorithme

Chaque sommet v :

- choisit uniformément au hasard l'un de ses voisins, le $i$ème ;
- envoie 0 à tous ses voisins sauf au $i$ème à qui il envoie 1 ;

Exemple



$c(v_1) = v_2$
$c(v_2) = v_4$
$c(v_3) = v_4$
$c(v_4) = v_3$

Contexte

Intérêt

Pas d'algorithme distribué déterministe pour le problème du rendez-vous.

Plus généralement

Sous certaines hypothèses, il n'existe pas de solution déterministe de certains problèmes classiques de l'algorithmique distribuée.
Les solutions probabilistes sont alors nécessaires à la résolution de ces problèmes.

Pas d'algorithme distribué déterministe pour le problème du rendez-vous.

Plus généralement

Sous certaines hypothèses, il n'existe pas de solution déterministe de certains problèmes classiques de l'algorithmique distribuée.
Les solutions probabilistes sont alors nécessaires à la résolution de ces problèmes.

Certaines solutions probabilistes sont plus efficaces que les déterministes.

Pas d'algorithme distribué déterministe pour le problème du rendez-vous.

Plus généralement

Sous certaines hypothèses, il n'existe pas de solution déterministe de certains problèmes classiques de l'algorithmique distribuée.
Les solutions probabilistes sont alors nécessaires à la résolution de ces problèmes.

Certaines solutions probabilistes sont plus efficaces que les déterministes.

Deux champs de recherche :

- Analyser des algorithmes distribués probabilistes
  - Etude de la complexité (en temps, en espace, en nombre de messages, etc)
  - Propriétés
- Prouver la correction

Contexte

Travaux reliés

Algorithmes distribués

- Modèle : Calculs locaux [MS97]
- Bibliothèque Coq : Loco [CF10]

Algorithmes probabilistes

- Approche impérative : générateur de nombre aléatoire [Hur02]
- Approche fonctionnelle : distribution de probabilité [APM09]

Algorithmes distribués probabilistes

- Analyse de certains algorithmes distribués probabilistes ([Tel94],[Lav95],[MR95],[Zem09])

Plan

1. Contexte
2. Résultat principal
3. Modélisation en Coq
4. Preuves
5. Conclusion

Résultat principal

L'algorithme *papier* du rendez-vous

Algorithme *papier*

Chaque sommet v :

- choisit uniformément au hasard l'un de ses voisins, le $i$ème ;
- envoie 0 à tous ses voisins sauf au $i$ème à qui il envoie 1 ;

Définitions

- $\mathcal{HS}(e)$: évènement "il y a un rendez-vous sur l'arête e".
- $\mathcal{H}(e)$: fonction caractéristique de $\mathcal{HS}(e)$

Résultat principal

$$\mathbb{P}(\exists\, e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-1/2} \approx 0,39$$

En moyenne, pour avoir un rendez-vous, il faut 3 rondes.

Résultat principal

L'algorithme *papier* du rendez-vous

Algorithme *papier*

Chaque sommet v :

- choisit uniformément au hasard l'un de ses voisins, le $i$ème ;
- envoie 0 à tous ses voisins sauf au $i$ème à qui il envoie 1 ;

Définitions

- $\mathcal{HS}(e)$: évènement "il y a un rendez-vous sur l'arête e".
- $\mathcal{H}(e)$: fonction caractéristique de $\mathcal{HS}(e)$

Résultat principal

$$\mathbb{P}(\exists\, e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-1/2} \approx 0,39$$

En moyenne, pour avoir un rendez-vous, il faut 3 rondes.

Résultat principal

L'algorithme *papier* du rendez-vous

Algorithme *papier*

Chaque sommet v :

- choisit uniformément au hasard l'un de ses voisins, le $i$ème ;
- envoie 0 à tous ses voisins sauf au $i$ème à qui il envoie 1 ;

Définitions

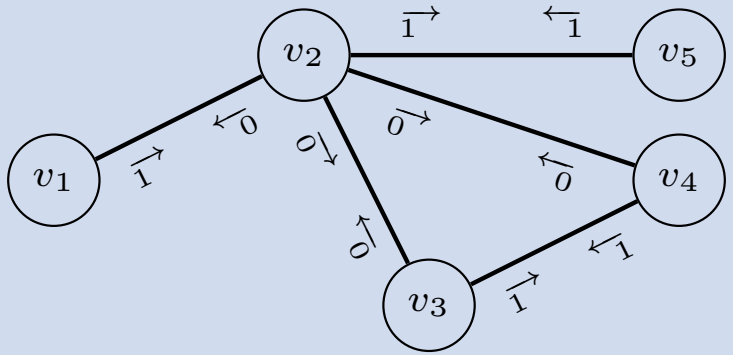
- $\mathcal{HS}(e)$: évènement "il y a un rendez-vous sur l'arête e".
- $\mathcal{H}(e)$: fonction caractéristique de $\mathcal{HS}(e)$

Résultat principal

$$\mathbb{P}(\exists\ e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-1/2} \approx 0,39$$

En moyenne, pour avoir un rendez-vous, il faut 3 rondes.

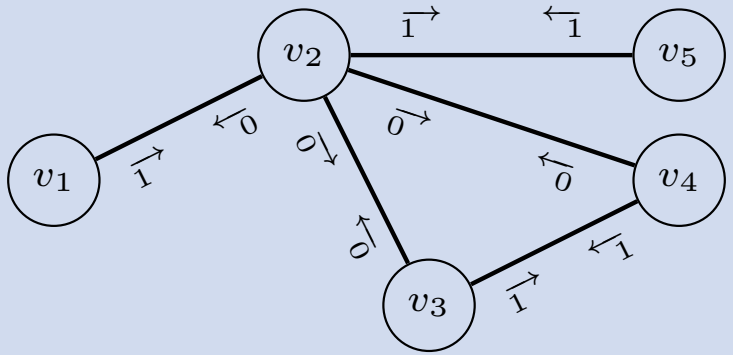
Exemple d'une ronde



Définition : Zone d'influence d'un sommet v

La *zone d'influence* d'un sommet v est l'ensemble des ports dont il peut modifier les étiquettes.

Exemple d'une ronde



Définition : Zone d'influence d'un sommet v

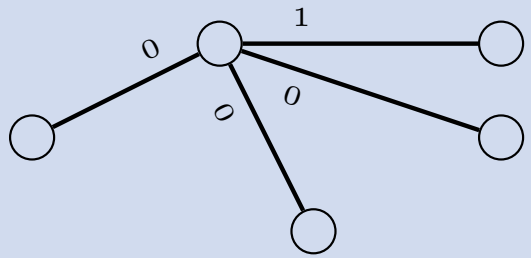
La *zone d'influence* d'un sommet v est l'ensemble des ports dont il peut modifier les étiquettes.

Plan

1. Contexte
2. Résultat principal
3. Modélisation en Coq
4. Preuves
5. Conclusion

Algorithme

```
(* Fhs_local : V → State → State *)
     let Fhs_local v σ =
       let k = random (d v)-1 in
         write_ports v k σ
```

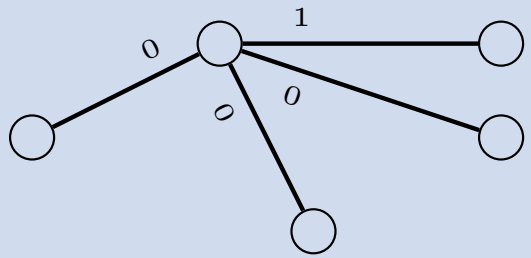


Modélisation en Coq

L'algorithme en $\mathcal{R}ml$ [APM09] du rendez-vous

Algorithme

```
(* Fhs_local : V → State → State *)
     let Fhs_local v σ =
       let k = random (d v)-1 in
         write_ports v k σ
```



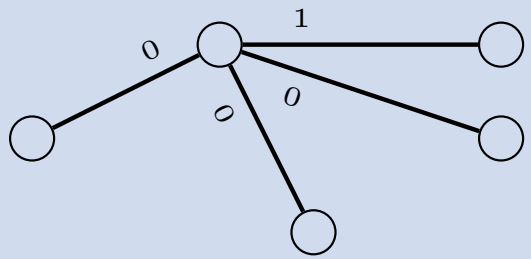

```
(* Fround : list V → State → State *)
     let rec Fround sV σ =
       if (null sV) then σ
       else let r = Fround (tail sV) σ in
         Fhs_local (head sV) r
```

Modélisation en Coq

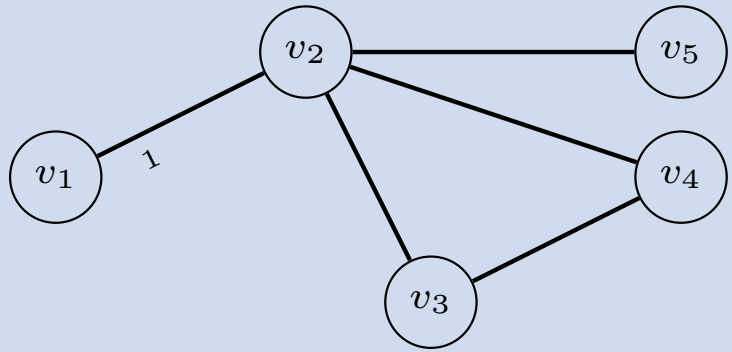
L'algorithme en $\mathcal{R}ml$ [APM09] du rendez-vous

Algorithme

```
(* Fhs_local : V → State → State *)
     let Fhs_local v σ =
       let k = random (d v)-1 in
         write_ports v k σ
```




```
(* Fround : list V → State → State *)
     let rec Fround sV σ =
       if (null sV) then σ
       else let r = Fround (tail sV) σ in
         Fhs_local (head sV) r
```



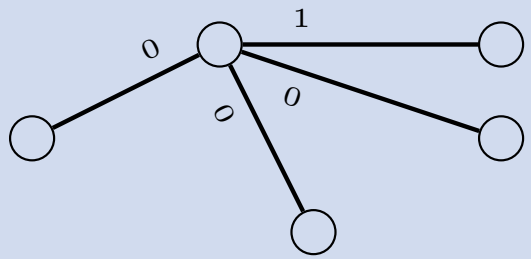

```
(* Fhs : graph →State → State *)
     let Fhs (V,E) σ =
       Fround (enum V) σ
```

Modélisation en Coq

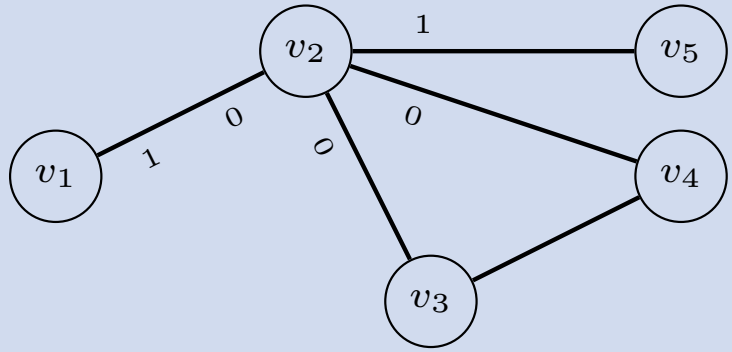
L'algorithme en $\mathcal{R}ml$ [APM09] du rendez-vous

Algorithme

```
(* Fhs_local : V → State → State *)
     let Fhs_local v σ =
       let k = random (d v)-1 in
         write_ports v k σ
```




```
(* Fround : list V → State → State *)
     let rec Fround sV σ =
       if (null sV) then σ
       else let r = Fround (tail sV) σ in
         Fhs_local (head sV) r
```



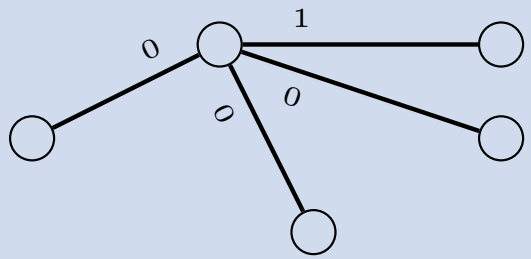

```
(* Fhs : graph →State → State *)
     let Fhs (V,E) σ =
       Fround (enum V) σ
```

Modélisation en Coq

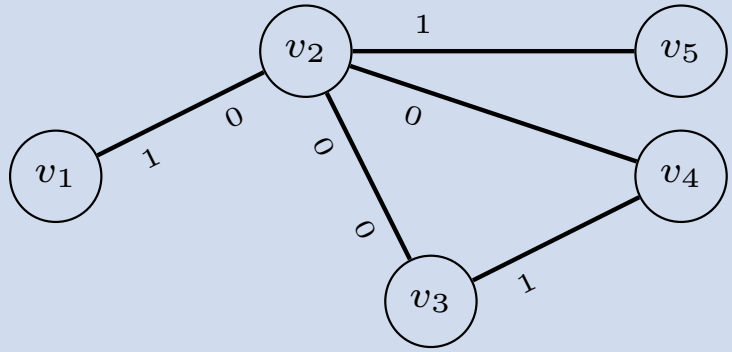
L'algorithme en $\mathcal{R}ml$ [APM09] du rendez-vous

Algorithme

```
(* Fhs_local : V → State → State *)
     let Fhs_local v σ =
       let k = random (d v)-1 in
         write_ports v k σ
```




```
(* Fround : list V → State → State *)
     let rec Fround sV σ =
       if (null sV) then σ
       else let r = Fround (tail sV) σ in
         Fhs_local (head sV) r
```



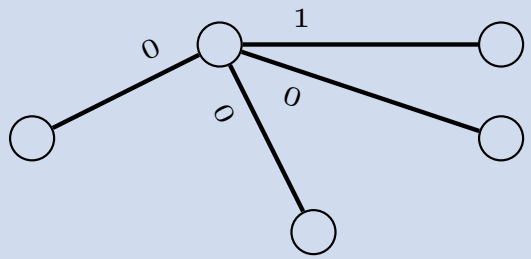

```
(* Fhs : graph →State → State *)
     let Fhs (V,E) σ =
       Fround (enum V) σ
```

Modélisation en Coq

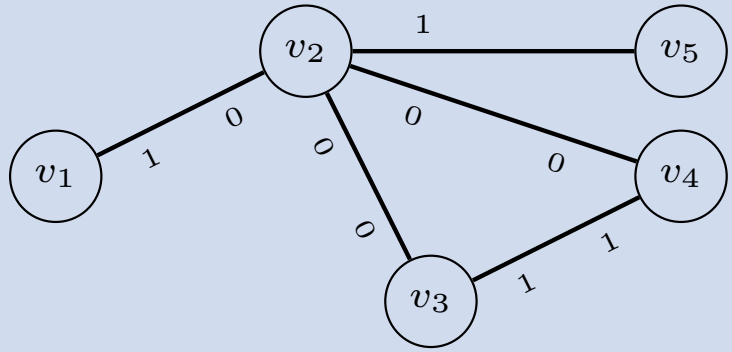
L'algorithme en $\mathcal{R}ml$ [APM09] du rendez-vous

Algorithme

```
(* Fhs_local : V → State → State *)
     let Fhs_local v σ =
       let k = random (d v)-1 in
         write_ports v k σ
```




```
(* Fround : list V → State → State *)
     let rec Fround sV σ =
       if (null sV) then σ
       else let r = Fround (tail sV) σ in
         Fhs_local (head sV) r
```



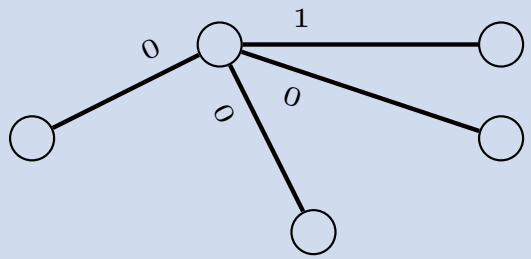

```
(* Fhs : graph →State → State *)
     let Fhs (V,E) σ =
       Fround (enum V) σ
```

Modélisation en Coq

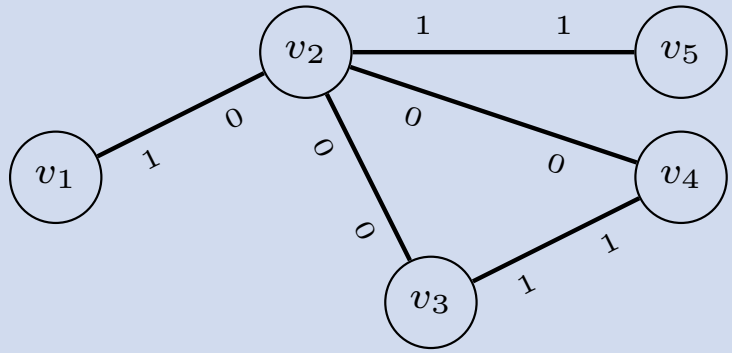
L'algorithme en $\mathcal{R}ml$ [APM09] du rendez-vous

Algorithme

```
(* Fhs_local : V → State → State *)
     let Fhs_local v σ =
       let k = random (d v)-1 in
         write_ports v k σ
```




```
(* Fround : list V → State → State *)
     let rec Fround sV σ =
       if (null sV) then σ
       else let r = Fround (tail sV) σ in
         Fhs_local (head sV) r
```

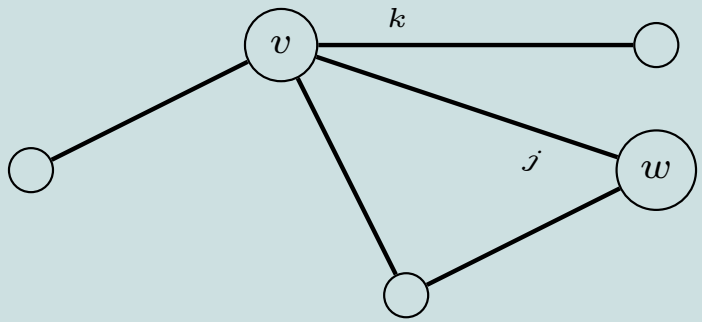



```
(* Fhs : graph →State → State *)
     let Fhs (V,E) σ =
       Fround (enum V) σ
```

Aspect distribué

- Lemme *(labelling.update_Pcomm)*
  Si $v \neq w$ alors
  $(\texttt{write\_ports}\ v\ k\ (\texttt{write\_ports}\ w\ j\ \sigma)) = (\texttt{write\_ports}\ w\ j\ (\texttt{write\_ports}\ v\ k\ \sigma))$

- Lemme *(FGlobalCommute3)*
  Soit lv une liste de sommets de G. Soit lv' une permutation de lv, nous avons :
  $$\texttt{Fround}\ lv\ \sigma = \texttt{Fround}\ lv'\ \sigma$$



Aspect probabiliste [APM09]

- Définition *(Probabilité)*
  Soit Q une propriété.
  $[e]\mathbb{1}_Q$: probabilité pour le résultat de l'expression e de satisfaire Q.
- Exemple :
  `[random n] = fun (f:nat → [0,1])` $\Rightarrow \sum_{i=0}^{n} \frac{1}{1+n}(f\ i)$

[random 5]

$$[\texttt{random 5}]\ f = \sum_{i=0}^{5} \frac{1}{6}(f\ i)$$

$$\mathbb{1}_{.=2\vee.=4} \begin{cases} 0 \mapsto 0 \\ 1 \mapsto 0 \\ 2 \mapsto 1 \\ 3 \mapsto 0 \\ 4 \mapsto 1 \\ 5 \mapsto 0 \end{cases}$$

$$\begin{aligned} [\texttt{random 5}]\ \mathbb{1}_{.=2\vee.=4} &= \sum_{i=0}^{5} \frac{1}{6}(\mathbb{1}_{i=2\vee i=4}\ ) \\ &= \frac{1}{6}*0+\frac{1}{6}*0+\frac{1}{6}*1+\frac{1}{6}*0+\frac{1}{6}*1+\frac{1}{6}*0 \\ &= \frac{1}{3} \end{aligned}$$

Aspect probabiliste [APM09]

- Définition *(Probabilité)*
  Soit Q une propriété.
  $[e]\mathbb{1}_Q$: probabilité pour le résultat de l'expression e de satisfaire Q.
- Exemple :
  `[random` n`] = fun (`f`:nat` $\to$ `[0,1]) ` $\Rightarrow \sum_{i=0}^{n} \frac{1}{1+n}(f\ i)$

[random 5]

`[random 5]` f `=` $\sum_{i=0}^{5} \frac{1}{6}(f\ i)$

$$\mathbb{1}_{.=2\vee.=4} \begin{cases} 0 \mapsto 0 \\ 1 \mapsto 0 \\ 2 \mapsto 1 \\ 3 \mapsto 0 \\ 4 \mapsto 1 \\ 5 \mapsto 0 \end{cases}$$

`[random 5]` $\mathbb{1}_{.=2\vee.=4}$

$= \sum_{i=0}^{5} \frac{1}{6}(\mathbb{1}_{i=2\vee i=4}\ )$

$= \frac{1}{6} * 0 + \frac{1}{6} * 0 + \frac{1}{6} * 1 + \frac{1}{6} * 0 + \frac{1}{6} * 1 + \frac{1}{6} * 0$

$= \frac{1}{3}$

Plan

1. Contexte
2. Résultat principal
3. Modélisation en Coq
4. **Preuves**
5. Conclusion

Preuves

Un premier résultat

Théorème

$\forall\ G\ \sigma\ \{v, w\}$, `[Fround` $sV\ \sigma$`]`$\mathbb{1}_{P(v,w)}$`=` $1/d(v)$.
où $P(v, w)$ est la propriété *"v choisit w"*

Lemme *handshake.Fhs_total*

$\forall\ \sigma\ s$, `[Fround` $s\ \sigma$`]`$\mathbb{1}$`= 1`

Idée de la preuve

Etape de récurrence sur s.

- Hypothèse : propriété vraie pour s'
- On veut montrer que c'est vrai pour (v :: s') :
  $\forall\ \sigma\ s'\ v$, `[Fround` $v :: s'\ \sigma$`]`$\mathbb{1}$ `= 1`
  $\forall\ \sigma\ s'\ v$, `[let x = (Fhs_local` $v\ \sigma$`) in (Fround` s' `x)]`$\mathbb{1}$ `= 1`
  $\forall\ \sigma\ s'\ v$, `[Fhs_local` $v\ \sigma$`] (fun x` $\Rightarrow$ `[Fround` s' `x]`$\mathbb{1}$`) = 1`
  $\forall\ \sigma\ v$, `[Fhs_local` $v\ \sigma$`]`$\mathbb{1}$ `= 1`

Preuves

Un premier résultat

Théorème

$\forall\ G\ \sigma\ \{v, w\}$, [Fround $sV\ \sigma$]$\mathbb{1}_{P(v,w)}$= $1/d(v)$.
où $P(v, w)$ est la propriété *"v choisit w"*

Lemme *handshake.Fhs_total*

$\forall\ \sigma\ s$, [Fround $s\ \sigma$]$\mathbb{1}$= 1

Idée de la preuve

Etape de récurrence sur s.

- Hypothèse : propriété vraie pour s'
- On veut montrer que c'est vrai pour (v :: s') :
  $\forall\ \sigma\ s'\ v$, [Fround $v :: s'\ \sigma$]$\mathbb{1}$ = 1
  $\forall\ \sigma\ s'\ v$, [let x = (Fhs_local $v\ \sigma$) in (Fround s' x)]$\mathbb{1}$ = 1
  $\forall\ \sigma\ s'\ v$, [Fhs_local $v\ \sigma$] (fun x $\Rightarrow$ [Fround s' x]$\mathbb{1}$) = 1
  $\forall\ \sigma\ v$, [Fhs_local $v\ \sigma$]$\mathbb{1}$ = 1

Preuves

Un premier résultat

Théorème

$\forall\ G\ \sigma\ \{v, w\}$, [Fround $sV\ \sigma$]$\mathbb{1}_{P(v,w)}$= $1/d(v)$.
où $P(v, w)$ est la propriété *"v choisit w"*

Lemme *handshake.Fhs_total*

$\forall\ \sigma\ s$, [Fround $s\ \sigma$]$\mathbb{1}$= 1

Idée de la preuve

Etape de récurrence sur s.

- Hypothèse : propriété vraie pour s'
- On veut montrer que c'est vrai pour (v :: s') :
  $\forall\ \sigma\ s'\ v$, [Fround $v :: s'\ \sigma$]$\mathbb{1}$ = 1
  $\forall\ \sigma\ s'\ v$, [let x = (Fhs_local $v\ \sigma$) in (Fround s' x)]$\mathbb{1}$ = 1
  $\forall\ \sigma\ s'\ v$, [Fhs_local $v\ \sigma$] (fun x $\Rightarrow$ [Fround s' x]$\mathbb{1}$) = 1
  $\forall\ \sigma\ v$, [Fhs_local $v\ \sigma$]$\mathbb{1}$ = 1

Preuves

Un premier résultat

Théorème

$\forall\ G\ \sigma\ \{v, w\}$, `[Fround` $sV\ \sigma$`]`$\mathbb{1}_{P(v,w)}$`=` $1/d(v)$.
où $P(v, w)$ est la propriété *"v choisit w"*

Idée de la preuve

$\forall\ G\ \sigma\ \{v, w\}$, `[Fround` $(v :: (sV \setminus v))\ \sigma$`]`$\mathbb{1}_{P(v,w)}$`=` $1/d(v)$

Preuves

Un premier résultat

Théorème

$\forall\ G\ \sigma\ \{v, w\}$, `[Fround` $sV\ \sigma$`]`$\mathbb{1}_{P(v,w)}$`=` $1/d(v)$.
où $P(v, w)$ est la propriété *"v choisit w"*

Idée de la preuve

$\forall\ G\ \sigma\ \{v, w\}$, `[Fround (`$v$ `::` `(`$sV \backslash v$`)) `σ`]`$\mathbb{1}_{P(v,w)}$`=` $1/d(v)$

Preuves

Preuve du résultat principal

Théorème

$$\mathbb{P}(\exists\ e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-\frac{1}{2}}$$

Lemme *my_alea.Mcond_prodConjBound*

$\forall i \in 1..m, \forall e \in E, \quad \mathbb{P}(\mathcal{H}(e)) \leq \mathbb{P}(\mathcal{H}(e) | \prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)})$

Idée de la preuve

- $\forall i \in 1..m,\ \mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}) \neq 0$
-

Preuves

Preuve du résultat principal

Théorème

$$\mathbb{P}(\exists\ e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-\frac{1}{2}}$$

Lemme *my_alea.Mcond_prodConjBound*

$\forall i \in 1..m, \forall e \in E, \quad \mathbb{P}(\mathcal{H}(e)) \leq \mathbb{P}(\mathcal{H}(e) | \prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)})$

Idée de la preuve

- $\forall i \in 1..m,\ \mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}) \neq 0$
-

Preuves

Preuve du résultat principal

Théorème

$$\mathbb{P}(\exists\ e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-\frac{1}{2}}$$

Lemme *my_alea.Mcond_prodConjBound*

$\forall i \in 1..m, \forall e \in E, \quad \mathbb{P}(\mathcal{H}(e)) \leq \mathbb{P}(\mathcal{H}(e) | \prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)})$

Idée de la preuve

- $\forall i \ \in 1..m,\ \mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}) \neq 0$
- Partition des arêtes :
  $S = \{e_{i+1}, \ldots, e_m\}$
  A : arêtes adjacentes à e
  B : celles qui ne le sont pas

$$\mathbb{P}(\mathcal{H}(e)) \leq \frac{\mathbb{P}(\mathcal{H}(e) * A * B)}{\mathbb{P}(A * B)}$$

$$= \frac{\mathbb{P}(\mathcal{H}(e) * \mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}))}{\mathbb{P}(\mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}))}$$

Lemme *my_alea.proba_not_null*

Soit A un algorithme probabiliste et E un évènement.

S'il existe un témoin t t.q.

$$\underbrace{[A]\mathbb{1}_{.=t} > 0}_{\textit{exécution possible}} \quad \text{et} \quad \underbrace{(E\ t) > 0}_{\textit{évènement vérifié}}$$

alors $\underbrace{[A]\mathbb{1}_{(E\ .)} > 0}_{\textit{A réalise E}}$

Preuves

Preuve du résultat principal

Théorème

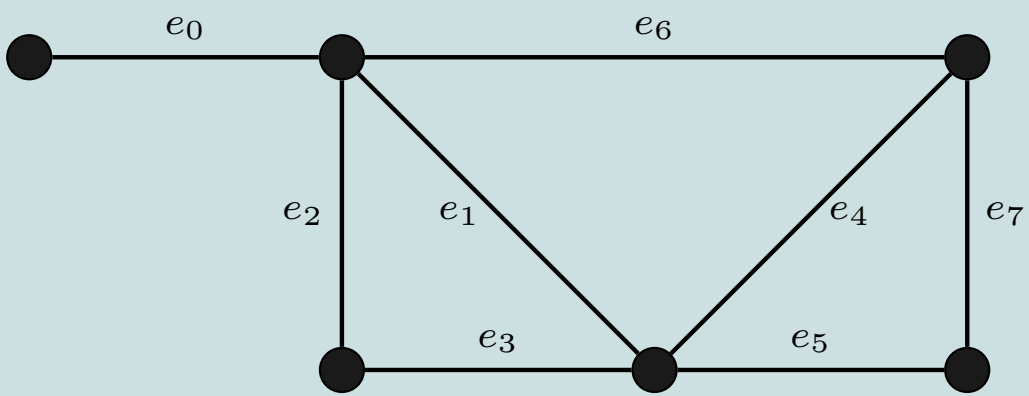
$$\mathbb{P}(\exists\ e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-\frac{1}{2}}$$

Lemme *handshake.hs1*

$$\forall i,\ \mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}) \neq 0$$

Idée de la preuve

Témoin : Arbre dont la racine est une extrémité de e_0



Théorème

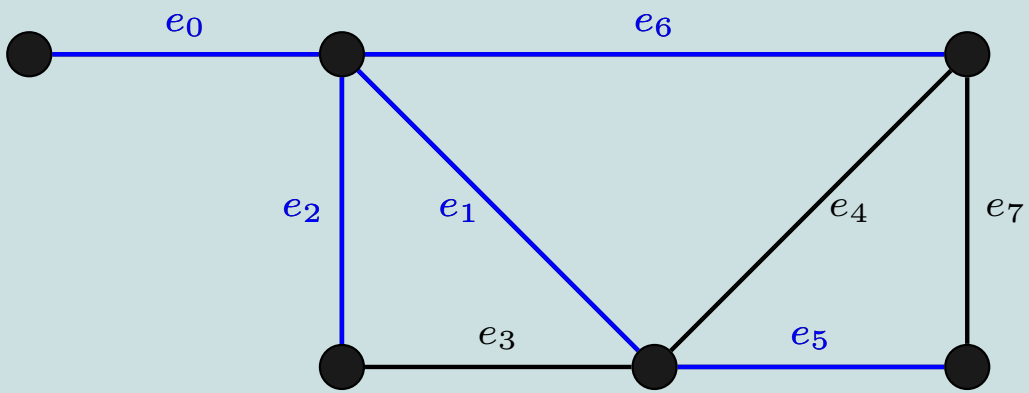
$$\mathbb{P}(\exists\ e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-\frac{1}{2}}$$

Lemme *handshake.hs1*

$$\forall i,\ \mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}) \neq 0$$

Idée de la preuve

Témoin : Arbre dont la racine est une extrémité de e_0



Théorème

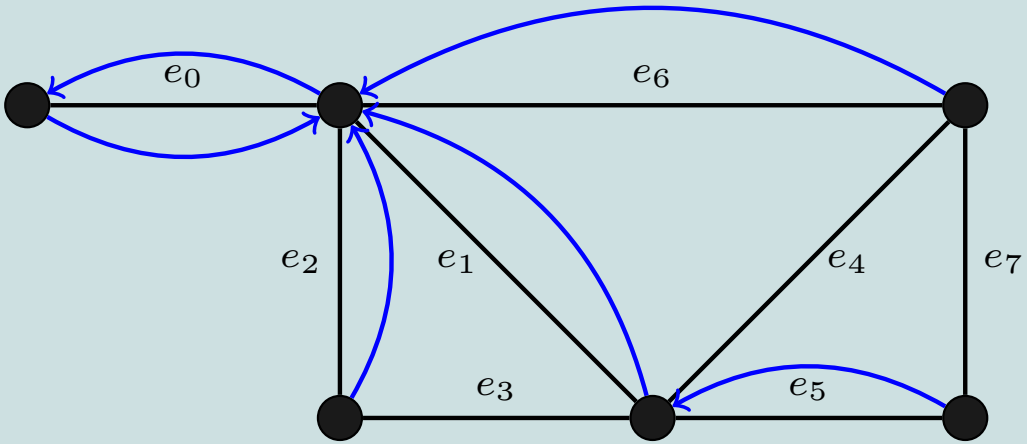
$$\mathbb{P}(\exists\ e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-\frac{1}{2}}$$

Lemme *handshake.hs1*

$$\forall i,\ \mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}) \neq 0$$

Idée de la preuve

Témoin : Arbre dont la racine est une extrémité de e_0



Preuves

Preuve du résultat principal

Théorème

$$\mathbb{P}(\exists\ e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-\frac{1}{2}}$$

Lemme *my_alea.Mcond_prodConjBound*

$\forall i \in 1..m, \forall e \in E, \quad \mathbb{P}(\mathcal{H}(e)) \leq \mathbb{P}(\mathcal{H}(e) | \prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)})$

Idée de la preuve

- $\forall i\ \in 1..m,\ \mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}) \neq 0$
- Partition des arêtes :
  $S = \{e_{i+1}, \ldots, e_m\}$

Preuves

Preuve du résultat principal

Théorème

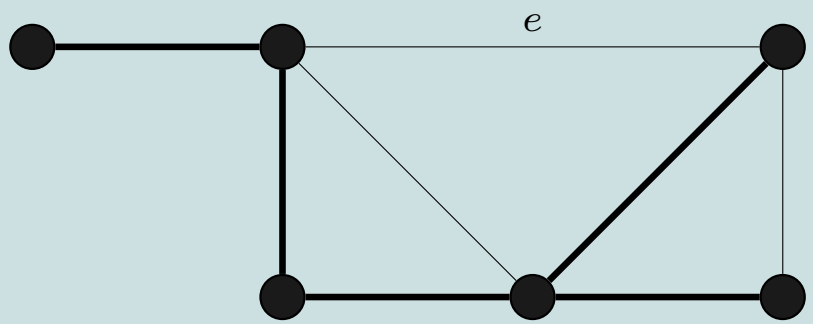
$$\mathbb{P}(\exists\ e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-\frac{1}{2}}$$

Lemme *my_alea.Mcond_prodConjBound*

$\forall i \in 1..m, \forall e \in E, \quad \mathbb{P}(\mathcal{H}(e)) \leq \mathbb{P}(\mathcal{H}(e) | \prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)})$

Idée de la preuve

- $\forall i\ \in 1..m,\ \mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}) \neq 0$
- Partition des arêtes :
  $\mathbf{S} = \{e_{i+1}, \ldots, e_m\}$



Preuves

Preuve du résultat principal

Théorème

$$\mathbb{P}(\exists\ e \in E,\ \mathcal{H}(e)) \geq 1 - e^{-\frac{1}{2}}$$

Lemme *my_alea.Mcond_prodConjBound*

$\forall i \in 1..m, \forall e \in E, \quad \mathbb{P}(\mathcal{H}(e)) \leq \mathbb{P}(\mathcal{H}(e) | \prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)})$

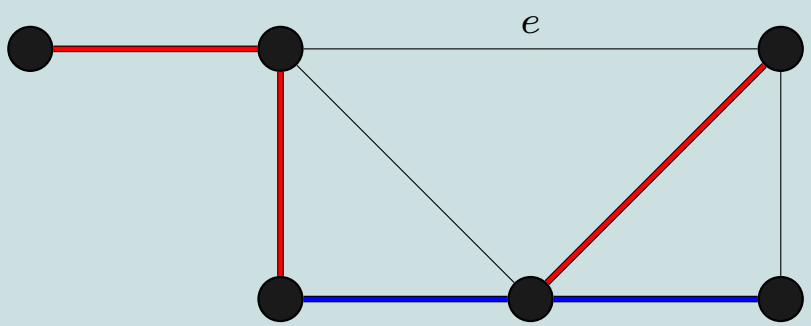
Idée de la preuve

- $\forall i\ \in 1..m,\ \mathbb{P}(\prod_{j=i+1}^{m} \overline{\mathcal{H}(e_j)}) \neq 0$
- Partition des arêtes :
  $\mathbf{S} = \{e_{i+1}, \ldots, e_m\}$
  A : arêtes adjacentes à e
  B : celles qui ne le sont pas

$$1 \leq \mathbb{P}(B) \backslash \mathbb{P}(A * B)$$

$$\mathbb{P}(\mathcal{H}(e)) \leq \frac{\mathbb{P}(\mathcal{H}(e) * B)}{\mathbb{P}(A * B)}$$

$$\mathbb{P}(\mathcal{H}(e)) \leq \frac{\mathbb{P}(\mathcal{H}(e) * A * B)}{\mathbb{P}(A * B)}$$



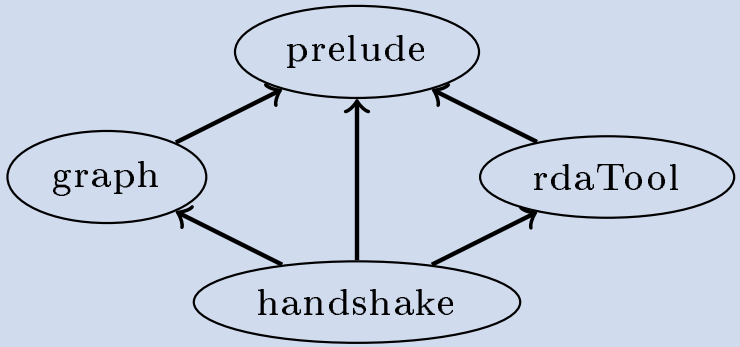
Plan

1. Contexte
2. Résultat principal
3. Modélisation en Coq
4. Preuves
5. Conclusion

Conclusion

Structure

```
(* Fround : list V → State → State *)
     let rec Fround sV σ =
       if (null sV) then σ
       else let r = Fround (tail sV) σ in
         Fhs_local (head sV) r
```



Expérience

- Au départ : Preuve mathématique
  - Modélisation
- Alea : Manipulation de l'intervalle $[0, 1]$
  - Pas de dépassement de capacité
  - Peu de simplification automatique
  - Ajout de nouveaux lemmes

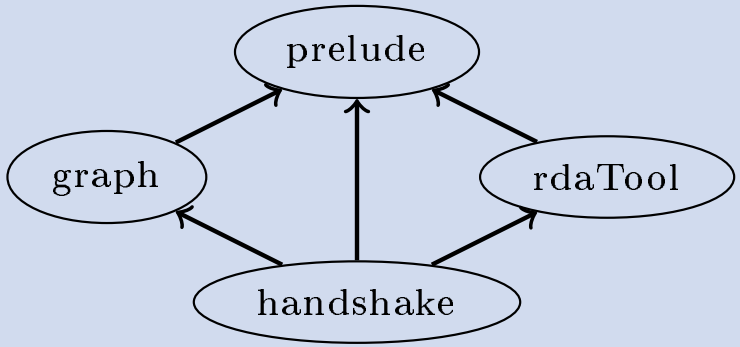
Perspectives

- Tactiques
- Couplage maximal
  - Itération de l'algorithme pour le rendez-vous
  - Exécution infinie possible
- Impossibilité probabiliste

Conclusion

Structure

```
(* Fround : list V → State → State *)
     let rec Fround sV σ =
       if (null sV) then σ
       else let r = Fround (tail sV) σ in
         Fhs_local (head sV) r
```



Expérience

- Au départ : Preuve mathématique
  - Modélisation
- Alea : Manipulation de l'intervalle $[0, 1]$
  - Pas de dépassement de capacité
  - Peu de simplification automatique
  - Ajout de nouveaux lemmes

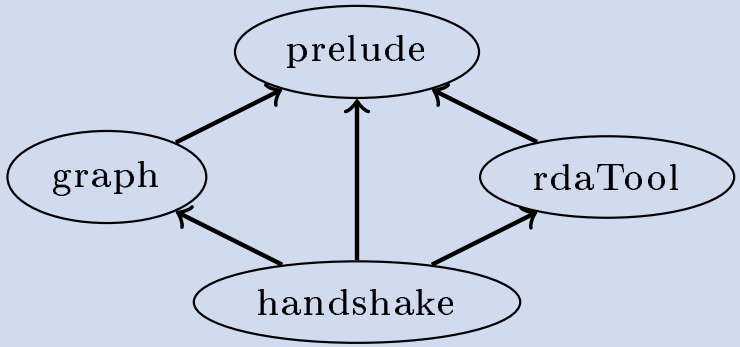
Perspectives

- Tactiques
- Couplage maximal
  - Itération de l'algorithme pour le rendez-vous
  - Exécution infinie possible
- Impossibilité probabiliste

Conclusion

Structure

```
(* Fround : list V → State → State *)
     let rec Fround sV σ =
       if (null sV) then σ
       else let r = Fround (tail sV) σ in
         Fhs_local (head sV) r
```



Expérience

- Au départ : Preuve mathématique
  - Modélisation
- Alea : Manipulation de l'intervalle $[0, 1]$
  - Pas de dépassement de capacité
  - Peu de simplification automatique
  - Ajout de nouveaux lemmes

Perspectives

- Tactiques
- Couplage maximal
  - Itération de l'algorithme pour le rendez-vous
  - Exécution infinie possible
- Impossibilité probabiliste

Bibliography I

Philippe Audebaud and Christine Paulin-Mohring.

Proofs of randomized algorithms in coq.
Sci. Comput. Program., 74(8) :568–589, 2009.

Pierre Castéran and Vincent Filou.

Tâches, types et tactiques pour les systèmes de calculs locaux.
In *Journées Francophones des Langages Applicatifs*, 2010.

Joe Hurd.

Formal Verification of Probabilistic Algorithms.
PhD thesis, University of Cambridge, 2002.

Christian Lavault.

Evaluation des algorithmes distribués – Analyse, complexité, méthodes.
Hermès, 1995.

Rajeev Motwani and Prabhakar Raghavan.

Randomized algorithms.
Cambridge University Press, New York, NY, USA, 1995.

Yves Métivier and Eric Sopena.

Graph relabelling systems : A general overview.
Computers and Artificial Intelligence, 16(2), 1997.

Gerard Tel.

Introduction to distributed algorithms.
Cambridge University Press, New York, NY, USA, 1994.

Akka Zemmari.

Présentation et analyse de quelques algorithmes distribués probabilistes.
Habilitation à Diriger les Recherches de l'Université Bordeaux 1 (spécialité : informatique), 2009.