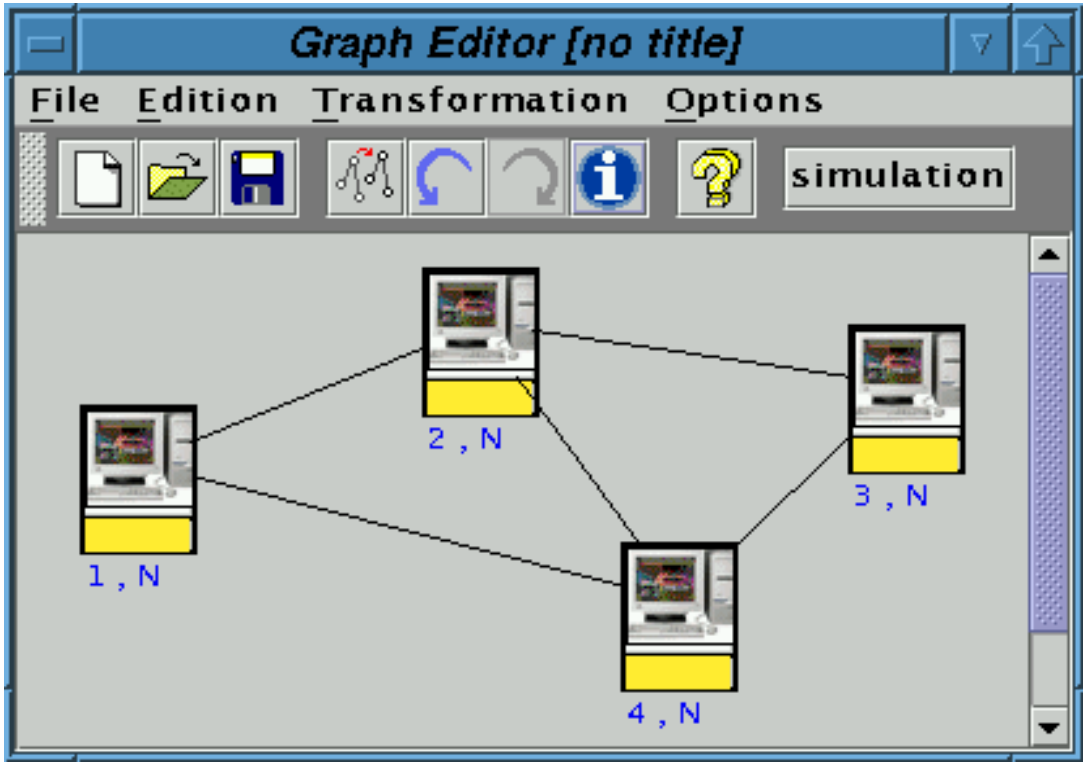
LaBRI



Présentation du projet Visidia

Mohamed Mosbah

mosbah@labri.fr

http://www.labri.fr/visidia/

- Construction d’algorithmes « efficaces » sur des classes de graphes « particulières »
- Elaboration de méthodes générales
- Modèle fondée sur les transformations de graphes (réécriture de graphes, des sommets et des arêtes, des étiquettes)
- Réalisation de plate-forme de tests et d’expérimentation ….

Algorithmes distribués

- Les algorithmes distribués sont complexes : communications entre processus, états des processus, synchronisation, terminaison, pas de temps universel.
- Pas de modèle universel pour les algorithmes distribués (synchrone vs asynchrone, mémoire partagée vs messages, etc)
- Difficulté de les étudier, les comparer et surtout de les implémenter

Objectifs:

- Élaborer un modèle général pour exprimer et décrire les algorithmes et les systèmes distribués

- Mettre au point un langage de description des transformations de graphes

- Réaliser une plate-forme pour implémenter les algorithmes distribués, et valider le modèle.

Fondement théorique

- Exemples de problèmes: Election d'un chef dans un réseau, Nommage, Calcul distribué d'un arbre recouvrant, Détection de la terminaison, Synchronisation, Consensus, etc.

- Un problème + une famille de réseaux + un modèle :

Questions: Existe-t-il un algorithme distribué permettant de le résoudre ? Sinon quelles sont les hypothèses nécessaires ? Peut-on caractériser la famille de graphes pour laquelle il existe une solution ?

→ Frontière entre calculable et non-calculable en algorithmique distribuée (selon les hypothèses ou la connaissance)

- Implantation ? Déterministe, probabiliste. Efficacité, performance…

Applications

- Compréhension et explication des algorithmes distribués
- Enseignement
- Aide aux preuves et mise au point
- Expérimentation et analyse des algorithmes distribués

- Les systèmes de réécriture (d'étiquettes) de graphes : modèle pour décrire les calculs distribués (Y. Métivier et al.)

- Autres modèles proches: Modèles à états (Dijkstra), IOA(Lynch), Rosenstiehl, Angluin, Yamashita et Kameda, Boldi et Vigna

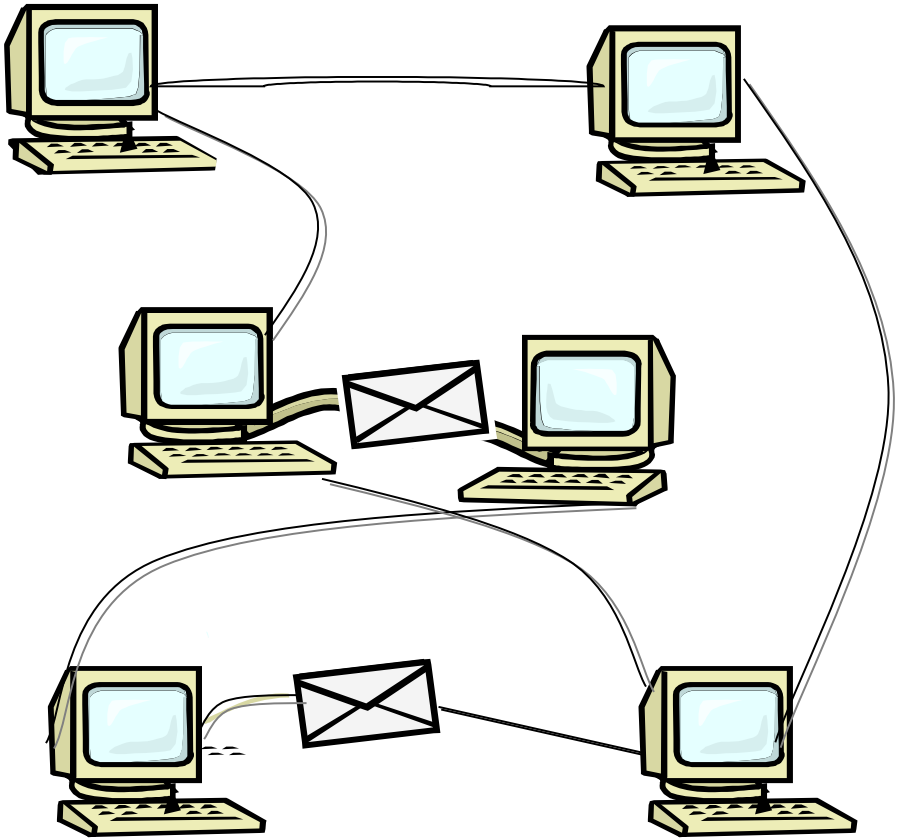
Visidia

- Principe :
  - Coder l'état d'un process ou d'un canal de communication par une étiquette
  - Transition d'un état à un autre : réécriture d'une étiquette
    - En général, la transition dépend de l'état d'un voisin (ou de tous les voisins) selon des règles

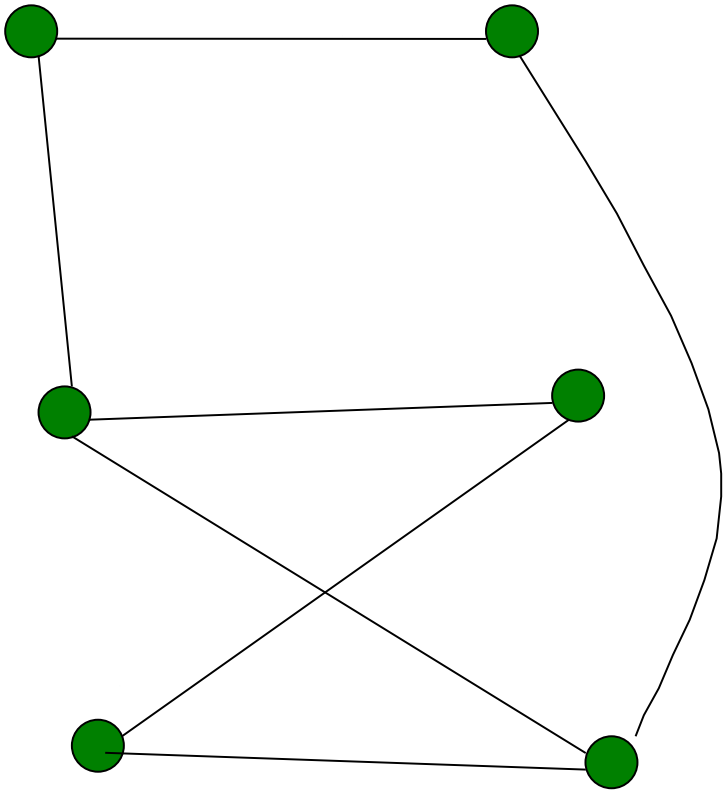
    → Réécriture d'étiquettes de graphes (deux sommets voisins, une boule, …)

Représentation :

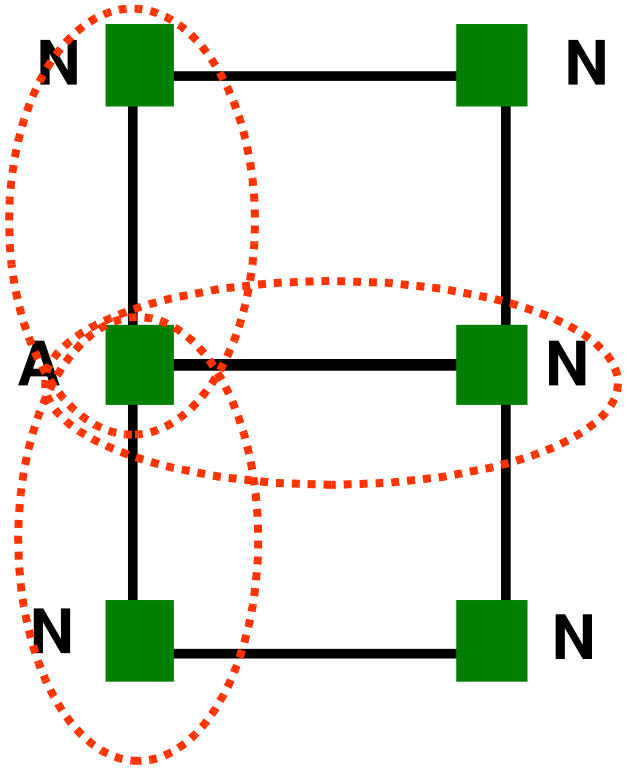
un réseau



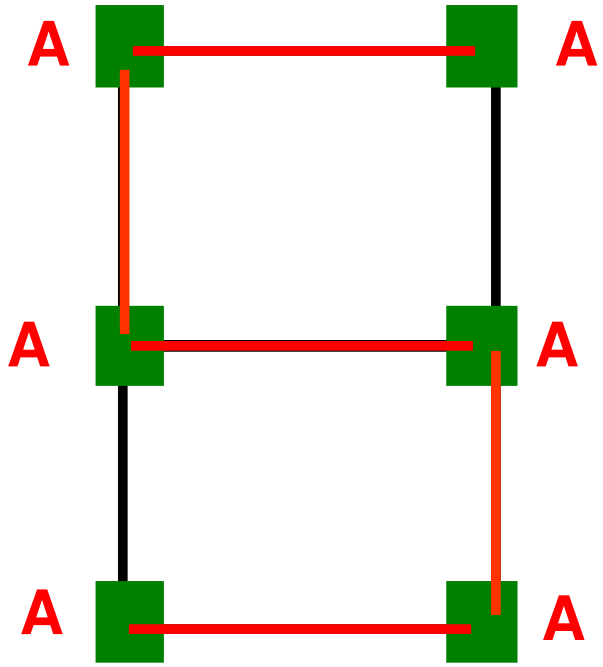
un graphe



Exemple



un arbre recouvrant



- Un système de réécriture est un triplet (L,I,P)
  - L : l'ensemble d'états (étiquettes)
  - I : l'ensemble d'états initiaux
  - P : l'ensemble des règles de réécritures R_i

Une étape de calcul : $(G,\lambda) \rightarrow (G, \lambda')$

(ne dépend que du contexte du sommet)

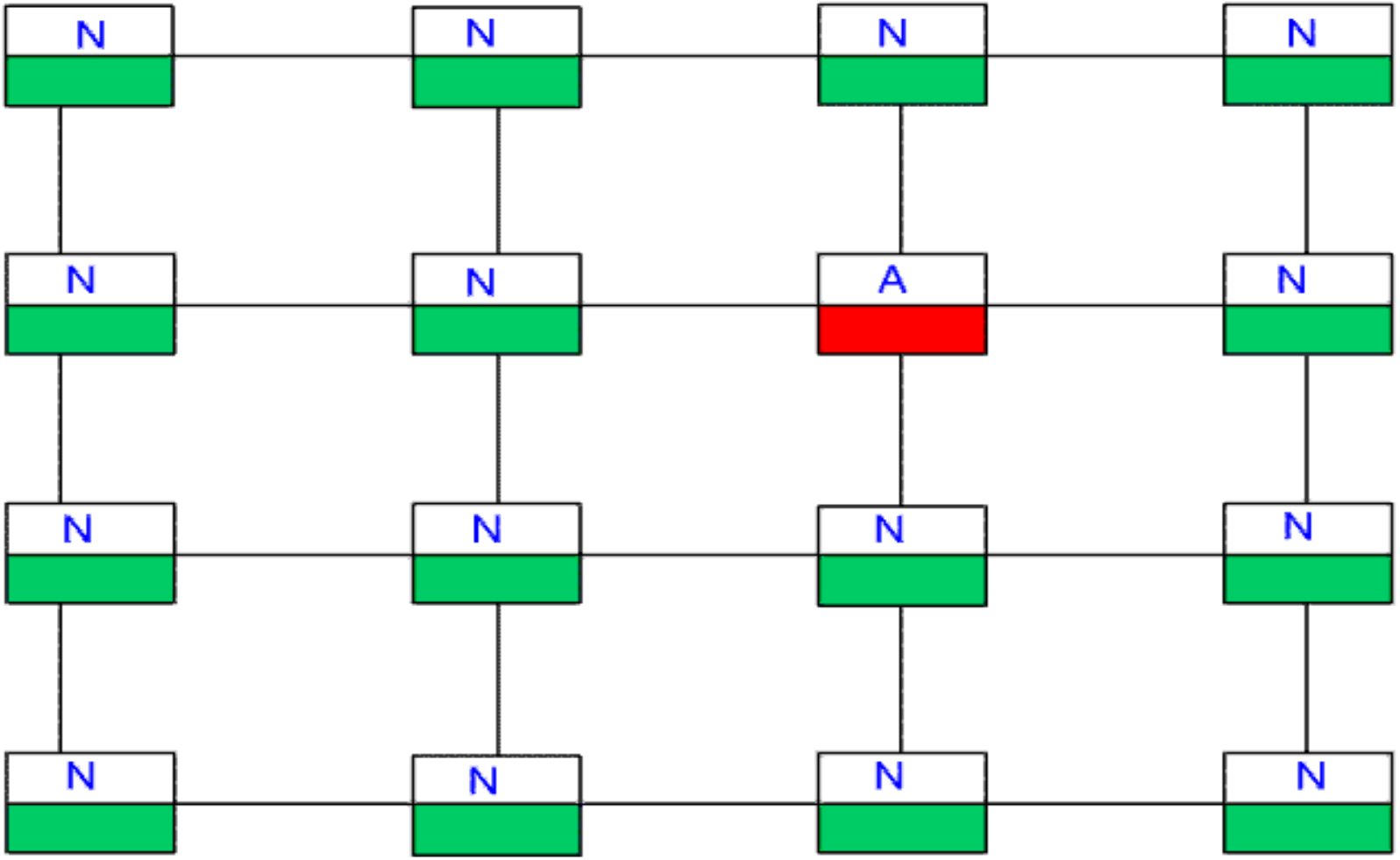
Structure de contrôle : priorité entre règles, contexte interdit

Codage d'algorithmes distribués

- Un sommet étiqueté A (Actif), et tous les autres sont étiquetés N (neutre)
- Règle:
  - Si un sommet x étiqueté A possède un voisin y étiqueté N
  - Alors x est étiqueté A et l'arête (x,y) est marquée

- Résultats : arbre recouvrant
- plusieurs règles peuvent être appliquées en //

Calcul distribué d'un arbre recouvrant



Preuves :

1/Terminaison : noethérien (pas de chaînes de réécritures infinies).

Technique: exhiber un ordre noethérien compatible avec le SR → trouver un ensemble partiellement ordonné (S,<) qui n'a pas de chaîne infinie décroissante et une fonction

f : GL → S / (G,λ) → (G,λ') alors f(G,λ) > f(G,λ')

2/ Correction : invariants (stables sur les chaînes de réécritures)

3/ Complexité : nombre de réécritures

Exemple (Arbre recouvrant avec SR1)

SR1 = (L1,I1,P1)

Théorème:

1. SR1 est noethérien (nombre de sommets N)
2. Soit (G,λ) un graphe étiqueté dans I1, avec un seul sommet étiqueté A, et soit (G,λ') un graphe SR1-irréductible. Le sous-graphe induit par les arêtes étiquetées 1 est un arbre de G.

- Invariants:
  - P1: Toutes les arêtes incidentes à un sommet N sont étiquetées 0
  - P2: Tout sommet étiqueté A est incident à au moins une arête étiquetée 1
  - P3: Le sous graphe induit par les arêtes étiquetées 1 est un arbre (sans cycle)

Quelques paradigmes « capturés » par les systèmes de réécriture

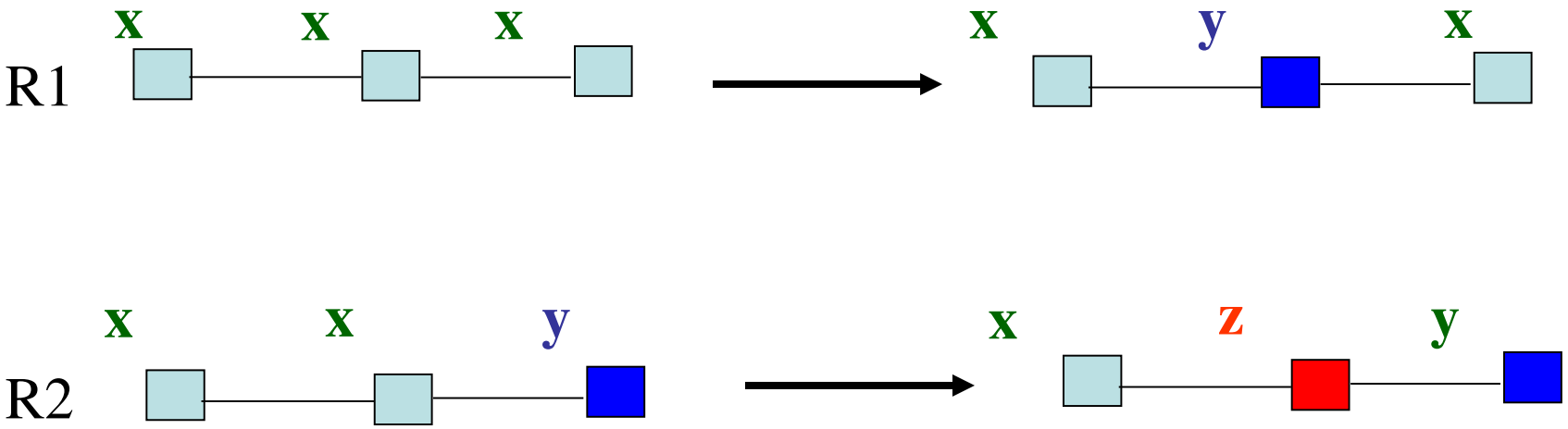
- Détection locale de la terminaison globale:

  L'algorithme est distribué ne *détecte pas localement la terminaison globale*.

- D'autres systèmes de réécriture permettent de capturer cette propriété.

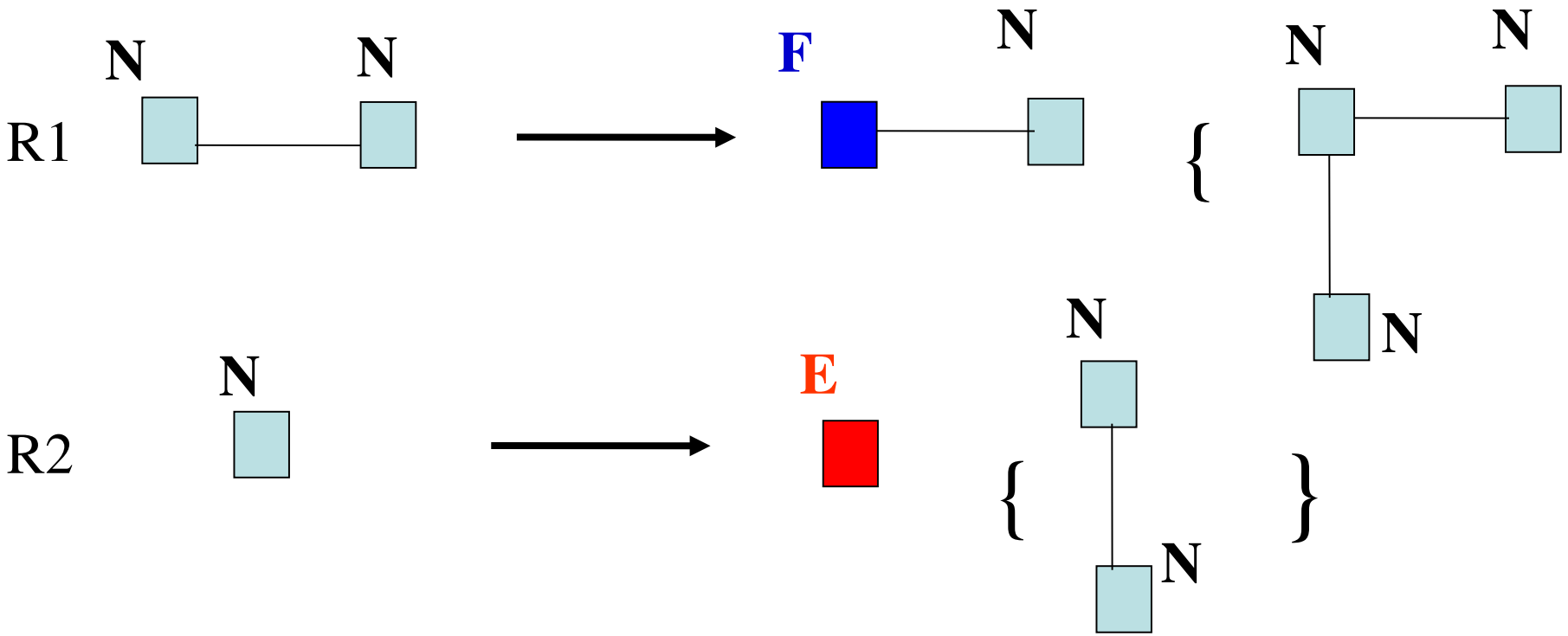
Un autre exemple: 3-coloration des sommets d'un anneau

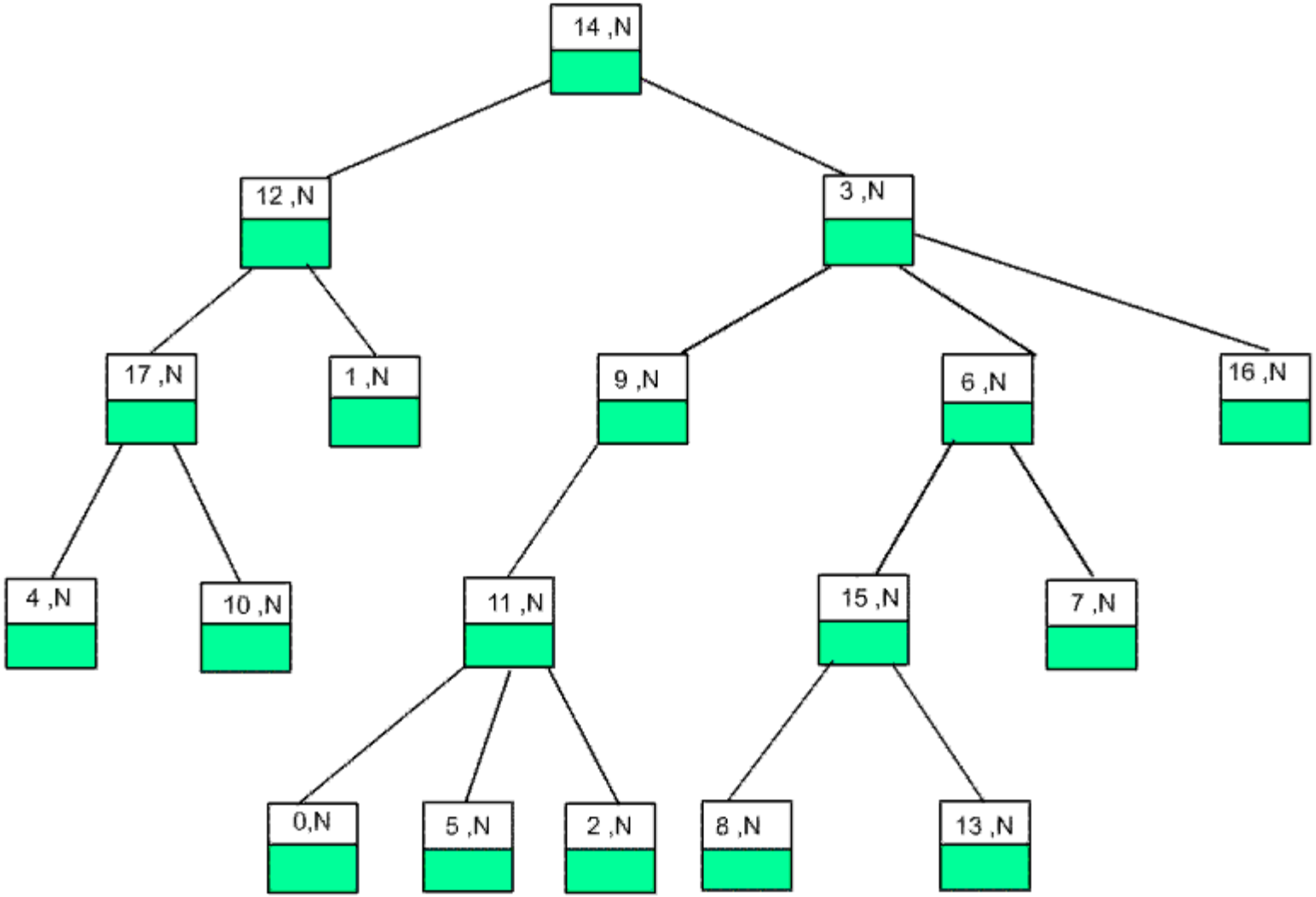
- Anneau avec des couleurs sur les sommets (initialement: configuration arbitraire)
- But: 3-coloration correcte (pas de sommets adjacents avec les mêmes couleurs).



Langage de description

- Expression des règles sous forme de precondition-action
- Exemple: Election




14 ,N
12 ,N
3 ,N
17 ,N
1 ,N
9 ,N
6 ,N
16 ,N
4 ,N
10 ,N
11 ,N
15 ,N
7 ,N
0,N
5 ,N
2 ,N
8 ,N
13 ,N

Precondition-relabeling rules

- At each node v, the following rules are executed:
- Pruning rule
  - Precondition /* v is a leaf */
    - λ (v) = N
    - ∃! neighbor u / λ (u) = N
  - Relabeling
    - λ (v) = F / * non-elected */
- Election rule
  - Precondition
    - λ (v) = N
    - ∀neighbor u / λ (u) ≠ N
  - Relabeling
    - λ (v) = E /* elected */

Formule logique

Lidia (aspects logiques)

- Langage de description d'algorithmes distribués
  - FOL avec
    - Variables indiv., constantes,
    - Quantificateurs: $\forall$, $\exists$,$\exists_i$, $\exists_i$
    - Connecteurs : non, et, ou,
    - Quelques fonctions arithmétiques: modulo, div, +,
    - Prédicats :

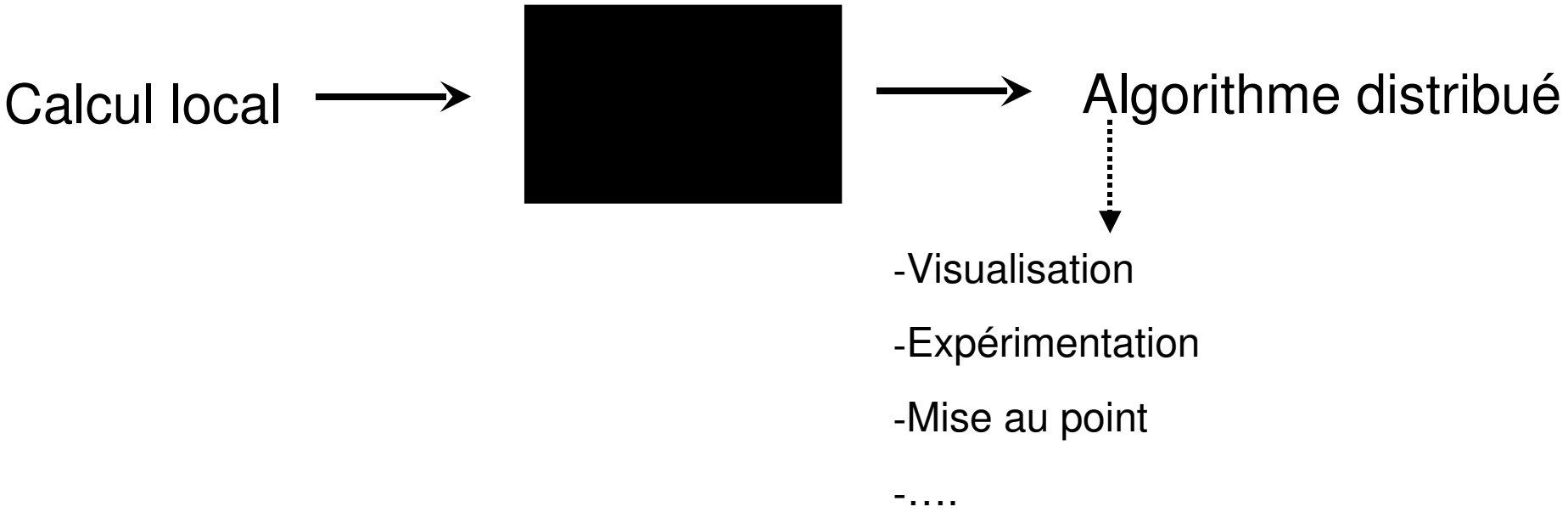
- **Exemple:** $\exists_1$ $v \in B(v0,1),\ v \neq v0,\ \lambda(u) = N$

Lidia

- Description « logique » de calculs locaux
  - Pré-condition (logique)
  - Action
- Caractérisation de la puissance des calculs locaux
- Langage de programmation

Présentation de visidia

- Méthode générale d'implantation (automatique) d'algorithmes distribués décrits par des calculs locaux



Program (Spanning Tree computation)

Declaration

type nLabel = tuple of var : string ;

type eLabel = tuple of var : int ;

G : graph<nLabel,eLabel> ;

edgeLab : eLabel ;

nodeLab : nLabel ;

B : node_set ;

L : node_array<nLabel> ;

v : node ;

Initialization

edgeLab.var := 0 ;

nodeLab.var := 'N' ;

G.init(nodeLab,edgeLab) ;

v :=G.choose_node() ;

v.label.var := 'A' ;

Synchronization

v_0

SpanningTree(v_0 : node) :

Universe

B := G.neighborhood(v_0) ;

Active rules : R_0 ;

L.init(B,nodeLab) ;

$\forall w \in B(w \neq v_0, IMPLIES L[w]$ *:=ReceiveFrom(w)) ;*

R_0 :={

Precondition

w : node ;

$v_0.label.\text{var} = 'N' \wedge \exists w \in B(w \neq v_0 \wedge L[w].\text{var} = 'A' \wedge [v_0, w].\text{var} = 0);$

Relabeling

$v_0.label.\text{var} = 'A';$

$[v_0, w].\text{var} := 1;$

Priorities

{} ;

} ;

Passive rules : P_0 ;

P_0 :={

Precondition

$v_0.sync \neq v_0;$

Action

v *: node ;*

$\forall v \in B(v \neq v_0 \wedge v.sync = v, IMPLIES\ SendTo(v,\ v_0.label))$ *;*

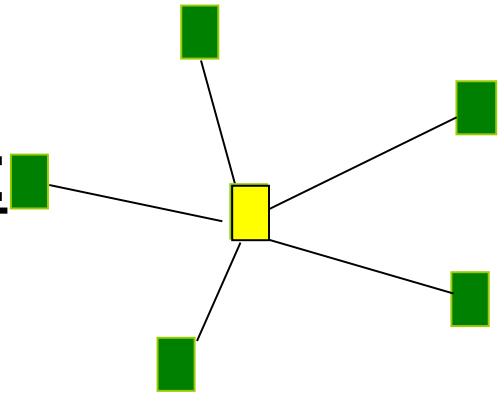
{} ;

SpanningTree.run(G) ;

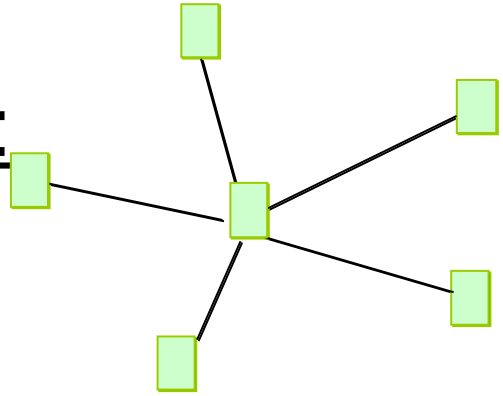
Types de règles

- Calcul local de type 0 (LC0) :

- Calcul local de type 1(LC1) :



- Calcul local de type 2 (LC2) :



Des règles de réécritures aux algorithmes distribués.

- Modèle du système distribué (général) :
  - sommet : entité autonome de calcul (processus, thread, etc) avec des portes numérotées,
  - arête : lien de communication entre deux ports,
  - communication : échange de messages via des ports,
  - primitives de base : *send, receive*,
  - réseau anonyme (ou avec identités).

Algorithme général

Chaque entité exécute les instructions suivantes :

```
While (run) {
\\ Synchronisation (selon le type de règle) ;
    \\ Echange d'étiquettes, attributs,
    \\ calculs ;
    \\ Mise à jour des étiquettes, attributs, états ;
\\ Arrêt de la  Synchronisation ;
}
```

Catalogue d'algorithmes distribués

Réseau anonyme:

1. Calcul distribué d'un arbre recouvrant sans détection locale de la terminaison
2. Calcul séquentiel d'un arbre recouvrant avec détection locale de la terminaison
3. Calcul distribué d'un arbre recouvrant avec détection de la terminaison
4. Élection dans un arbre
5. Élection un graphe complet
6. Algorithme de Mazurkiewicz de reconstruction universelle d'un graphe (énumération distribuée)
7. Algorithme de détection de propriétés stables (SSP)

1. 3-coloration d'un anneau sans détection locale de la terminaison
2. 3-coloration d'un anneau avec détection locale de la terminaison
3. d-coloration d'un graphe de degré borné par d-1
4. Algorithme de terminaison de Dijkstra-Scholten (arbre dynamique de nœuds actifs)
5. Algorithme de terminaison à l'aide d'un jeton
6. Gestion distribuée de conflits

Réseau avec identités:

8. Calcul distribué d'un arbre recouvrant sans détection de la terminaison
9. Calcul séquentiel d'un arbre recouvrant avec terminaison
10. Calcul distribué d'un arbre recouvrant avec terminaison
11. Élection dans un anneau avec identités (Chang Roberts)

Synchronisation locale

- Procédures probabilistes pour implémenter les synchronisations :
  - LC0 : implémenté par le rendez-vous
  - LC1 : procédure probabiliste d'élection locale dans une boule de rayon 1.
  - LC2 : procédure probabiliste d'élection locale dans une boule de rayon 2.

Outil logiciel visidia

- Construction d'un réseau (édition de graphes)
- Édition et/ou implémentation des règles de réécriture
- Association automatique d'un *thread* Java à chaque sommet
- Chaque entité exécute l'algorithme.

Travaux en cours:

- Etudier la validité de propriétés globales à partir de propriétés locales
- Caractériser les propriétés globales calculables

- Développer des outils facilitant la preuve de correction (en s'appuyant sur d'autres environnements)
  - Raffinement (B – événementiel)
  - Assistant de preuves (Coq)

  - Pour un algorithme donné
  - Pour une classe d'algorithmes

Etude des agents mobiles

- Un agent mobile est une entité mobile de calcul
- Exécution d'un algorithme distribué:
  - Réseau représentant un système distribué
  - Des agents se déplacent et exécutent les règles

  →modèle d'implantation de calculs locaux

  Etude de la puissance de ce modèle vs modèle classique

→Quelques paradigmes: synchronisation, modèle de déplacement, terminaison

→Extensions aux algorithmes distribués des réseaux de capteurs.