

Kleene et Kozen en Coq

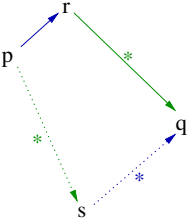
Thomas Braibant et Damien Pous (Grenoble)

Séminaire du LIX, 23 avril 2009

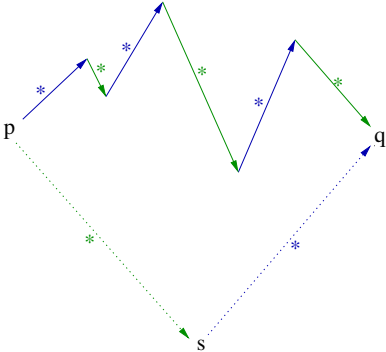
Motivations

- ▶ Faciliter la formalisation de certains types de preuves, en Coq.
- ▶ Initialement, des preuves de commutation, en réécriture abstraite.
- ▶ Exemple : “la confluence faible implique la propriété de Church-Rosser”.

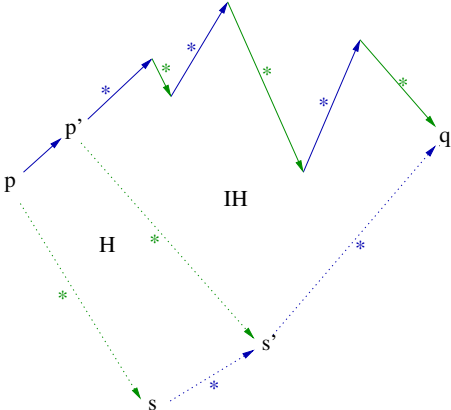
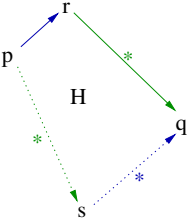
Church-Rosser



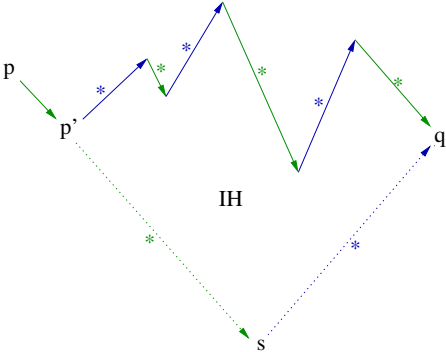
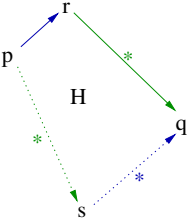
implique



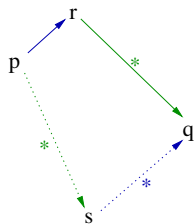
Church-Rosser (Preuve diagrammatique)



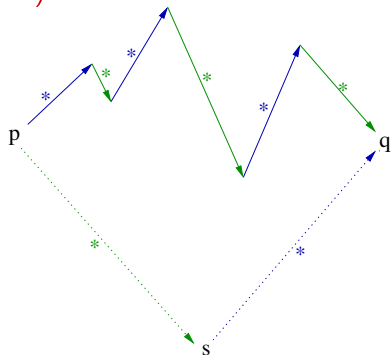
Church-Rosser (Preuve diagrammatique)



Church-Rosser (plus formellement)



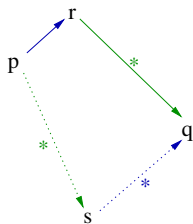
implique



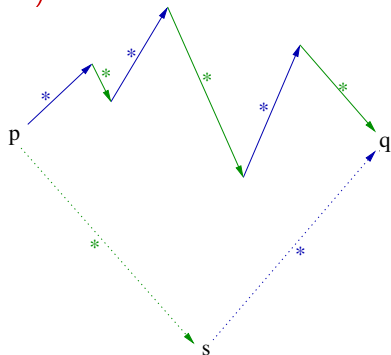
$$(\forall p, r, q, pRr, rS^*q \Rightarrow \exists s, pS^*s \wedge sR^*q)$$

$$\Rightarrow (\forall p, q, p(R \cup S)^*q \Rightarrow \exists s, pS^*s \wedge sR^*q)$$

Church-Rosser (plus formellement)



implique



$$(\forall p, r, q, pRr, rS^*q \Rightarrow \exists s, pS^*s \wedge sR^*q)$$

$$\Rightarrow (\forall p, q, p(R \cup S)^*q \Rightarrow \exists s, pS^*s \wedge sR^*q)$$

$$R \cdot S^* \subseteq S^* \cdot R^* \Rightarrow (R \cup S)^* \subseteq S^* \cdot R^*$$

Objectifs

- ▶ La présentation algébrique facilite les choses :
 - ▶ lisibilité des buts ;
- ▶ mais on a vu l'utilité :
 - ▶ de tactiques de décision (à la ring, omega) :
`kleene_reflexivity`, `monoid_reflexivity`,
`semiring_reflexivity`...
 - ▶ de tactiques de simplification (`ring_simplify`) :
`semiring_normalize`, `aci_normalize`...
 - ▶ de tactiques de réécriture (modulo A, modulo AC) :
`monoid_rewrite`, `ac_rewrite`.

Quelles algèbres ?

- ▶ Au final, on veut représenter des relations binaires (choix personnel : bisimulations, réécriture. . .).
- ▶ [Tarski et al.] : pas d'axiomatisation finie. On a néanmoins de nombreuses axiomatisations partielles :
 - ▶ monoïdes non commutatifs
 - ▶ treillis
 - ▶ semi-anneaux idempotents non commutatifs
 - ▶ algèbres de Kleene
 - ▶ semi-anneaux résiduels
 - ▶ algèbres d'actions (Pratt), treillis d'actions (Kozen)
 - ▶ algèbres de relations / allégories (Freyd & Scedrov)
- ▶ Selon les algèbres, plus ou moins de modèles, des propriétés de décidabilité ou de complexité différentes.

Plan

Motivations

Décision des algèbres de Kleene par les automates

- Rappel de la méthode

- Vision algébrique de l'algorithme

- Quelques détails d'implémentation Coq

Couches sous-jacentes du développement

- Algèbres, matrices

- Typage

- Effaçage des types

Conclusions et perspectives

Algèbres de Kleene

$(X, \cdot, +, 1, 0, \star)$

- ▶ $(X, \cdot, 1)$ est un monoïde, $(X, +, 0)$ est un semi-treillis,
- ▶ \cdot distribue à droite et à gauche sur $(+, 0)$,
- ▶ l'étoile de Kleene (\star) est définie par :

$$1 + x \cdot x^* \leq x^* \qquad \frac{a \cdot x \leq x}{a^* \cdot x \leq x} \qquad \frac{x \cdot a \leq x}{x \cdot a^* \leq x}$$

(où $x \leq y \triangleq x + y = y$).

Algèbres de Kleene

$(X, \cdot, +, 1, 0, \star)$

- ▶ $(X, \cdot, 1)$ est un monoïde, $(X, +, 0)$ est un semi-treillis,
- ▶ \cdot distribue à droite et à gauche sur $(+, 0)$,
- ▶ l'étoile de Kleene (\star) est définie par :

$$1 + x \cdot x^* \leq x^* \qquad \frac{a \cdot x \leq x}{a^* \cdot x \leq x} \qquad \frac{x \cdot a \leq x}{x \cdot a^* \leq x}$$

(où $x \leq y \triangleq x + y = y$).

- ▶ Deux modèles standard : les relations binaires, les expressions régulières.

Scott vs. Kozen

Soient α et β deux expressions régulières $(+, \cdot, 0, 1, ^*)$.

Scott '50 α et β désignent le même langage si et seulement si les automates minimaux correspondants sont isomorphes.

Scott vs. Kozen

Soient α et β deux expressions régulières $(+, \cdot, 0, 1, *)$.

Scott '50 α et β désignent le même langage si et seulement si les automates minimaux correspondants sont isomorphes.

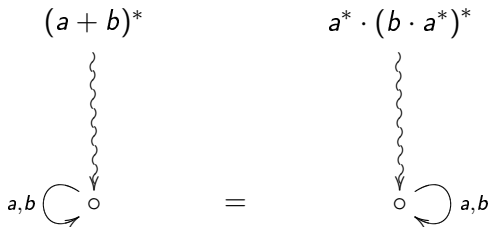
Kozen '94 Initialité de ce modèle pour les algèbres de Kleene :
Si α et β mènent au même automate,
alors $\mathcal{A} \vdash \alpha = \beta$, pour toute algèbre de Kleene \mathcal{A} .

Rappel : construction de l'automate minimal

On part d'une expression régulière ; on en dérive

1. un automate (non déterministe),
2. dont on doit retirer les ϵ -transitions,
3. afin de pouvoir le déterminer,
4. puis le minimiser.

Scott vs. Kozen (again)



Scott '50 : On en déduit $L((a + b)^*) = L(a^* \cdot (b \cdot a^*)^*)$.

Kozen '94 : On en déduit $\mathcal{A} \vdash (a + b)^* = a^* \cdot (b \cdot a^*)^*$.

En faire une tactique réflexive

- ▶ La complexité théorique de l'algo est exponentielle...
mais en pratique ça va...
- ▶ On veut fournir une tactique Coq utilisable (voire efficace).

En faire une tactique réflexive

- ▶ La complexité théorique de l'algo est exponentielle... mais en pratique ça va...
- ▶ On veut fournir une tactique Coq utilisable (voire efficace).
- ▶ Coq est un langage de programmation, on code raisonnablement l'algo :

```
Definition decide_Kleene : regexp → regexp → bool := ...
```

En faire une tactique réflexive

- ▶ La complexité théorique de l'algo est exponentielle... mais en pratique ça va...
- ▶ On veut fournir une tactique Coq utilisable (voire efficace).
- ▶ Coq est un langage de programmation, on code raisonnablement l'algo :

Definition decide_Kleene : regexp → regexp → bool := ...

- ▶ De manière à pouvoir faire la preuve de Kozen dans Coq :

Theorem Kozen : $\forall a b : \text{regexp},$

$$\text{decide_Kleene } a b = \text{true} \rightarrow \forall A, \langle a \rangle_A = \langle b \rangle_A.$$

- ▶ Puis on enrobe le tout dans une tactique.

Modèles initiaux

- ▶ Les mots finis sont le modèle initial des monoïdes
- ▶ Les ensembles finis sont le modèle initial des (semi)-treillis
- ▶ Les ensembles finis de mots finis sont le modèle initial des semi-anneaux idempotents
- ▶ ...

Modèles initiaux

- ▶ Les mots finis sont le modèle initial des monoïdes
- ▶ Les ensembles finis sont le modèle initial des (semi)-treillis
- ▶ Les ensembles finis de mots finis sont le modèle initial des semi-anneaux idempotents
- ▶ ...

Mais on ne connaît pas de preuve directe de l'initialité du modèle des langages réguliers (ensembles réguliers de mots finis) pour les algèbres de Kleene.

La preuve de Kozen

- L'idée de Kozen est de représenter les automates de façon algébrique, avec des matrices :

$$(\dots \quad u \quad \dots) \cdot \begin{pmatrix} \dots & \dots & \dots \\ \dots & M & \dots \\ \dots & \dots & \dots \end{pmatrix} \cdot \begin{pmatrix} \vdots \\ v \\ \vdots \end{pmatrix}$$

La preuve de Kozen

- ▶ L'idée de Kozen est de représenter les automates de façon algébrique, avec des matrices :

$$(\dots \quad u \quad \dots) \cdot \left(\begin{array}{ccc} \dots & \dots & \dots \\ \dots & M & \dots \\ \dots & \dots & \dots \end{array} \right)^* \cdot \begin{pmatrix} \vdots \\ v \\ \vdots \end{pmatrix}$$

- ▶ Les algèbres de Kleene sont stables par formation des matrices.

La preuve de Kozen

- ▶ L'idée de Kozen est de représenter les automates de façon algébrique, avec des matrices :

$$(\dots \quad u \quad \dots) \cdot \left(\begin{array}{ccc} \dots & \dots & \dots \\ \dots & M & \dots \\ \dots & \dots & \dots \end{array} \right)^* \cdot \left(\begin{array}{c} \vdots \\ v \\ \vdots \end{array} \right) =_{\mathcal{A}} \alpha$$

- ▶ Les algèbres de Kleene sont stables par formation des matrices.

La preuve de Kozen

- ▶ L'idée de Kozen est de représenter les automates de façon algébrique, avec des matrices :

$$(\dots \quad u \quad \dots) \cdot \left(\begin{array}{ccc} \dots & \dots & \dots \\ \dots & M & \dots \\ \dots & \dots & \dots \end{array} \right)^* \cdot \left(\begin{array}{c} \vdots \\ v \\ \vdots \end{array} \right) =_{\mathcal{A}} \alpha$$

- ▶ Les algèbres de Kleene sont stables par formation des matrices.
- ▶ Puis de retranscrire et valider les algorithmes de façon algébrique.

Séparation algorithme / preuves

$$(a + b)^* \equiv \alpha$$

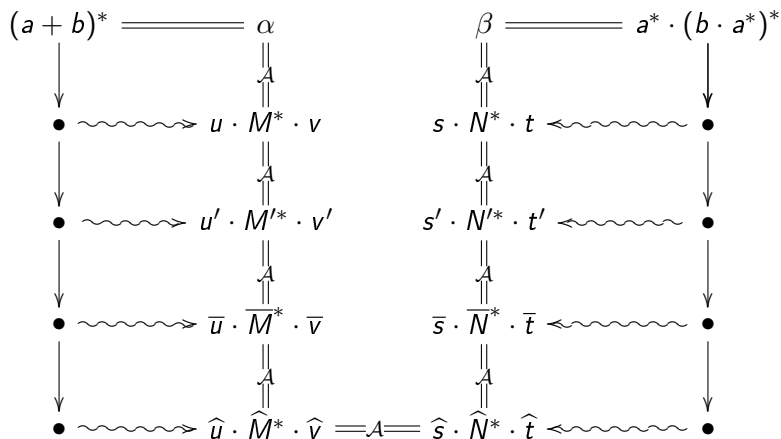
$$\beta \equiv a^* \cdot (b \cdot a^*)^*$$



Séparation algorithme / preuves

$$\begin{array}{ccc}
 (a + b)^* & \xlongequal{\quad} & \alpha & & \beta & \xlongequal{\quad} & a^* \cdot (b \cdot a^*)^* \\
 & & \parallel \mathcal{A} & & \parallel \mathcal{A} & & \\
 & & u \cdot M^* \cdot v & & s \cdot N^* \cdot t & & \\
 & & \parallel \mathcal{A} & & \parallel \mathcal{A} & & \\
 & & u' \cdot M'^* \cdot v' & & s' \cdot N'^* \cdot t' & & \\
 & & \parallel \mathcal{A} & & \parallel \mathcal{A} & & \\
 & & \bar{u} \cdot \bar{M}^* \cdot \bar{v} & & \bar{s} \cdot \bar{N}^* \cdot \bar{t} & & \\
 & & \parallel \mathcal{A} & & \parallel \mathcal{A} & & \\
 & & \hat{u} \cdot \hat{M}^* \cdot \hat{v} & \xlongequal{\mathcal{A}} & \hat{s} \cdot \hat{N}^* \cdot \hat{t} & &
 \end{array}$$

Séparation algorithme / preuves



Construction de Thomson (Plus)

Construction de l'automate correspondant à $\alpha + \beta$

► Si $\alpha = u \cdot M^* \cdot v$ et $\beta = s \cdot N^* \cdot t$

► Alors

$$\begin{aligned} [u \mid s] \cdot \left[\begin{array}{c|c} M & 0 \\ \hline 0 & N \end{array} \right]^* \cdot \left[\begin{array}{c} v \\ t \end{array} \right] \\ &= [u \mid s] \cdot \left[\begin{array}{c|c} M^* & 0 \\ \hline 0 & N^* \end{array} \right] \cdot \left[\begin{array}{c} v \\ t \end{array} \right] \\ &= \dots \\ &= u \cdot M^* \cdot v + s \cdot N^* \cdot t \\ &= \alpha + \beta \end{aligned}$$

Construction de Thomson (Dot)

Construction de l'automate correspondant à $\alpha \cdot \beta$

► Si $\alpha = u \cdot M^* \cdot v$ et $\beta = s \cdot N^* \cdot t$

► Alors

$$\begin{aligned} & \left[\begin{array}{c|c} u & 0 \end{array} \right] \cdot \left[\begin{array}{c|c} M & v \cdot s \\ \hline 0 & N \end{array} \right]^* \cdot \left[\begin{array}{c} 0 \\ t \end{array} \right] \\ &= \left[\begin{array}{c|c} u & 0 \end{array} \right] \cdot \left[\begin{array}{c|c} M^* & M^* \cdot v \cdot s \cdot N^* \\ \hline 0 & N^* \end{array} \right] \cdot \left[\begin{array}{c} 0 \\ t \end{array} \right] \\ &= \dots \\ &= u \cdot M^* \cdot v \cdot s \cdot N^* \cdot t \\ &= \alpha \cdot \beta \end{aligned}$$

Construction de Thomson (Star)

Construction de l'automate correspondant à α^*

▶ Si $\alpha = u \cdot M^* \cdot v$

▶ Alors

$$\begin{aligned}u \cdot (M + v \cdot u)^* \cdot v &= u \cdot M^* \cdot (v \cdot u \cdot M^*)^* \cdot v \\ &= u \cdot M^* \cdot v \cdot (u \cdot M^* \cdot v)^* \\ &= \alpha \cdot \alpha^*\end{aligned}$$

(on conclut avec $\alpha^* = 1 + \alpha \cdot \alpha^*$)

Construction de Thomson (Star)

Construction de l'automate correspondant à α^*

- ▶ Si $\alpha = u \cdot M^* \cdot v$
- ▶ Alors

$$\begin{aligned}u \cdot (M + v \cdot u)^* \cdot v &= u \cdot M^* \cdot (v \cdot u \cdot M^*)^* \cdot v \\ &= u \cdot M^* \cdot v \cdot (u \cdot M^* \cdot v)^* \\ &= \alpha \cdot \alpha^*\end{aligned}$$

(on conclut avec $\alpha^* = 1 + \alpha \cdot \alpha^*$)

- ▶ Pas d'étoile dans $M + v \cdot u$!

Élimination des ϵ -transitions

- ▶ En assemblant les constructions précédentes, on obtient des automates tels que

$$M = J + \sum_{a \in \Sigma} a \cdot M_a ,$$

où J et M_a sont des matrices 0-1.

- ▶ Un automate est ϵ -free si $J = 0$.
- ▶ Algo standard : calcul de la clôture réflexive transitive des ϵ -transitions.

Élimination des ϵ -transitions

- ▶ En assemblant les constructions précédentes, on obtient des automates tels que

$$M = J + \sum_{a \in \Sigma} a \cdot M_a ,$$

où J et M_a sont des matrices 0-1.

- ▶ Un automate est ϵ -free si $J = 0$.
- ▶ Algo standard : calcul de la clôture réflexive transitive des ϵ -transitions.
- ▶ Algébriquement, on a :

$$u \cdot (J + N)^* \cdot v = u \cdot J^* \cdot (N \cdot J^*)^* \cdot v .$$

Détermination

- ▶ Il s'agit de calculer l'automate des parties.
- ▶ Soit X la matrice de décodage de l'ensemble des parties accessibles pour l'automate (u, M, v) :

$$X_{sj} \triangleq j \in s$$

- ▶ On peut définir \bar{M} et \bar{u} tels que :

$$\bar{M} \cdot X = X \cdot M$$

$$\bar{u} \cdot X = u$$

Détermination

- ▶ Il s'agit de calculer l'automate des parties.
- ▶ Soit X la matrice de décodage de l'ensemble des parties accessibles pour l'automate (u, M, v) :

$$X_{sj} \triangleq j \in s$$

- ▶ On peut définir \overline{M} et \overline{u} tels que :

$$\overline{M}^* \cdot X = X \cdot M^* \qquad \overline{u} \cdot X = u$$

Détermination

- ▶ Il s'agit de calculer l'automate des parties.
- ▶ Soit X la matrice de décodage de l'ensemble des parties accessibles pour l'automate (u, M, v) :

$$X_{sj} \triangleq j \in s$$

- ▶ On peut définir \overline{M} et \overline{u} tels que :

$$\overline{M}^* \cdot X = X \cdot M^* \qquad \overline{u} \cdot X = u$$

- ▶ On en déduit

$$\begin{aligned} \overline{u} \cdot \overline{M}^* \cdot X \cdot v &= \overline{u} \cdot X \cdot M^* \cdot v \\ &= u \cdot M^* \cdot v \end{aligned}$$

Minimisation

- ▶ On doit fusionner les états 'équivalents'.
- ▶ Soit Y la matrice de décodage des classes d'équivalence pour l'automate (u, M, v) :

$$Y_{ij} \triangleq [i] = j$$

- ▶ On peut définir \widehat{M} et \widehat{v} tels que :

$$Y \cdot \widehat{M} = M \cdot Y \qquad Y \cdot \widehat{v} = v$$

- ▶ On en déduit

$$\begin{aligned} u \cdot Y \cdot \widehat{M}^* \cdot \widehat{v} &= u \cdot M^* \cdot Y \cdot \widehat{v} \\ &= u \cdot M^* \cdot v \end{aligned}$$

Isomorphisme d'automates

- ▶ Une fois calculés les automates minimaux de deux expressions, il faut décider si ces automates sont isomorphes.
- ▶ Matrice de permutation ?

Isomorphisme d'automates

- ▶ Une fois calculés les automates minimaux de deux expressions, il faut décider si ces automates sont isomorphes.
- ▶ Matrice de permutation ? Oui, mais c'est inefficace.

Isomorphisme d'automates

- ▶ Une fois calculés les automates minimaux de deux expressions, il faut décider si ces automates sont isomorphes.
- ▶ Matrice de permutation ? Oui, mais c'est **inefficace**.
- ▶ Etant donnés les automates déterminisés de α et β , il suffit de tester l'équivalence de leurs états initiaux dans l'automate somme.

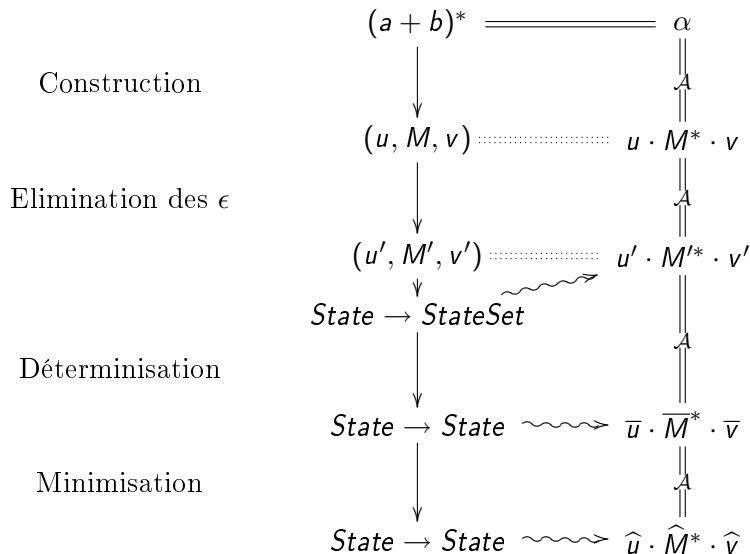
Isomorphisme d'automates

- ▶ Une fois calculés les automates minimaux de deux expressions, il faut décider si ces automates sont isomorphes.
- ▶ Matrice de permutation ? Oui, mais c'est inefficace.
- ▶ Etant donnés les automates déterminisés de α et β , il suffit de tester l'équivalence de leurs états initiaux dans l'automate somme.
- ▶ Algébriquement, quand ces états coïncident, on obtient :

$$\alpha + \beta = \alpha + 0 = \alpha$$

$$\alpha + \beta = 0 + \beta = \beta$$

Structures de données



Matrices

► Fonctions infinies :

Definition $MX\ n\ m := \text{nat} \rightarrow \text{nat} \rightarrow X.$

Definition $\text{equal}\ n\ m\ (M\ N : MX\ n\ m) :=$
 $\forall\ i\ j, i < n \rightarrow j < m \rightarrow M\ i\ j = N\ i\ j.$

Matrices

- ▶ Fonctions infinies :

Definition $MX\ n\ m := \text{nat} \rightarrow \text{nat} \rightarrow X.$

Definition $\text{equal}\ n\ m\ (M\ N : MX\ n\ m) :=$
 $\forall\ i\ j, i < n \rightarrow j < m \rightarrow M\ i\ j = N\ i\ j.$

- ▶ Facile à manipuler (séparation programmes/preuves)

Matrices

- ▶ Fonctions infinies :

Definition $\text{MX } n \text{ m} := \text{nat} \rightarrow \text{nat} \rightarrow X.$

Definition $\text{equal } n \text{ m } (M \ N : \text{MX } n \text{ m}) :=$
 $\forall i \ j, i < n \rightarrow j < m \rightarrow M \ i \ j = N \ i \ j.$

- ▶ Facile à manipuler (séparation programmes/preuves)
- ▶ Représentation paresseuse :
 - ▶ Fonction pour forcer l'évaluation, et mémoriser la matrice dans une structure concrète :
$$\forall n \text{ m } (M : \text{MX } n \text{ m}), \text{mx_force } M = M.$$
 - ▶ On peut insérer cette identité aux endroits adéquats (multiplication, étoile, ...) et garder la représentation paresseuse ailleurs (somme, ...).

Automates matriciels

- ▶ La représentation naïve des automates comme des matrices sur \mathcal{A} est inefficace :
 - ▶ il y a trop de termes : $\begin{bmatrix} a^* & b \cdot c \\ (1+c)^* & 0 \end{bmatrix}$
 - ▶ les termes ne sont pas canoniques, on doit raisonner modulo les axiomes des algèbres de Kleene :

$$\begin{bmatrix} 0 & a+b \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0+0 & a \cdot 1 + b + c \cdot 0 \\ 0^* & 0 \cdot 0 \end{bmatrix}$$

Automates matriciels

- ▶ La représentation naïve des automates comme des matrices sur \mathcal{A} est inefficace :
 - ▶ il y a trop de termes : $\begin{bmatrix} a^* & b \cdot c \\ (1+c)^* & 0 \end{bmatrix}$
 - ▶ les termes ne sont pas canoniques, on doit raisonner modulo les axiomes des algèbres de Kleene :

$$\begin{bmatrix} 0 & a+b \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0+0 & a \cdot 1 + b + c \cdot 0 \\ 0^* & 0 \cdot 0 \end{bmatrix}$$

- ▶ Les matrices de transitions des automates sont toutes de la forme $J + \sum_a a \cdot M_a$, où J et les M_a sont des matrices 0-1.
- ▶ Il suffit de construire les matrices sur l'algèbre de Kleene formée par les booléens, et d'injecter dans \mathcal{A} lorsque c'est nécessaire.

Automates matriciels

- ▶ La représentation naïve des automates comme des matrices sur \mathcal{A} est inefficace :
 - ▶ il y a trop de termes : $\begin{bmatrix} a^* & b \cdot c \\ (1+c)^* & 0 \end{bmatrix}$
 - ▶ les termes ne sont pas canoniques, on doit raisonner modulo les axiomes des algèbres de Kleene :

$$\begin{bmatrix} 0 & a+b \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0+0 & a \cdot 1 + b + c \cdot 0 \\ 0^* & 0 \cdot 0 \end{bmatrix}$$

- ▶ Les matrices de transitions des automates sont toutes de la forme $J + \sum_a a \cdot M_a$, où J et les M_a sont des matrices 0-1.
- ▶ Il suffit de construire les matrices sur l'algèbre de Kleene formée par les booléens, et d'injecter dans \mathcal{A} lorsque c'est nécessaire.
- ▶ Code spécialisé pour le calcul sur les matrices booléennes.

Plan

Motivations

Décision des algèbres de Kleene par les automates

- Rappel de la méthode

- Vision algébrique de l'algorithme

- Quelques détails d'implémentation Coq

Couches sous-jacentes du développement

- Algèbres, matrices

- Typage

- Effaçage des types

Conclusions et perspectives

Algèbres (de Kleene)

$(X, \cdot, +, 1, 0, \star)$

- ▶ $(X, \cdot, 1)$ est un monoïde, $(X, +, 0)$ est un semi-treillis,
- ▶ \cdot distribue à droite et à gauche sur $(+, 0)$,
- ▶ l'étoile de Kleene (\star) est définie par :

$$1 + x \cdot x^* \leq x^* \qquad \frac{a \cdot x \leq x}{a^* \cdot x \leq x} \qquad \frac{x \cdot a \leq x}{x \cdot a^* \leq x}$$

(où $x \leq y \triangleq x + y = y$).

Algèbres

En Coq, on structure les définitions de la même manière, afin de pouvoir hériter des outils pour les monoïdes, les semi-treillis, les semi-anneaux, etc. . . On utilise pour ça les **Typeclasses**.

SemiLattice

<: SemiRing <: KleeneAlg <: . . .

Monoid

Matrices

[Standard] : les matrices carrées sur un semi-anneau forment un semi-anneau ;

[Kozen '94] : les matrices carrées sur une algèbre de Kleene forment une algèbre de Kleene.

Matrices

[Standard] : les matrices carrées sur un semi-anneau forment un semi-anneau ;

[Kozen '94] : les matrices carrées sur une algèbre de Kleene forment une algèbre de Kleene.

Problème :

- ▶ A différents niveaux, on a besoin de manipuler des matrices rectangulaires,
- ▶ et on aimerait avoir accès aux outils des semi-anneaux même lorsque l'on travaille sur ces matrices.

Matrices

[Standard] : les matrices carrées sur un semi-anneau forment un semi-anneau ;

[Kozen '94] : les matrices carrées sur une algèbre de Kleene forment une algèbre de Kleene.

Problème :

- ▶ A différents niveaux, on a besoin de manipuler des matrices rectangulaires,
- ▶ et on aimerait avoir accès aux outils des semi-anneaux même lorsque l'on travaille sur ces matrices.

Une solution :

- ▶ Généraliser les structures algébriques avec du **typage** (i.e., passer dans les catégories).

Typage

X : Type.	T : Type. X : $T \rightarrow T \rightarrow \text{Type}$.
dot : $X \rightarrow X \rightarrow X$. one : X .	dot : $\forall A B C, X A B \rightarrow X B C \rightarrow X A C$. one : $\forall A, X A A$.
plus : $X \rightarrow X \rightarrow X$. zero : X .	plus : $\forall A B, X A B \rightarrow X A B \rightarrow X A B$. zero : $\forall A B, X A B$.
star : $X \rightarrow X$.	star : $\forall A, X A A \rightarrow X A A$.
dot_neutral_left : $\forall x, \text{dot one } x = x$.	dot_neutral_left : $\forall A B (x: X A B), \text{dot one } x = x$.
...	...

Typage

$X: \text{Type.}$	$T: \text{Type.}$
$\text{dot}: X \rightarrow X \rightarrow X.$	$X: T \rightarrow T \rightarrow \text{Type.}$
$\text{one}: X.$	$\text{dot}: \forall A B C, X A B \rightarrow X B C \rightarrow X A C.$
$\text{plus}: X \rightarrow X \rightarrow X.$	$\text{one}: \forall A, X A A.$
$\text{zero}: X.$	$\text{plus}: \forall A B, X A B \rightarrow X A B \rightarrow X A B.$
$\text{star}: X \rightarrow X.$	$\text{zero}: \forall A B, X A B.$
$\text{dot_neutral_left}: \forall x, \text{dot one } x = x.$	$\text{star}: \forall A, X A A \rightarrow X A A.$
\dots	$\text{dot_neutral_left}: \forall A B (x: X A B), \text{dot one } x = x.$
\dots	\dots

SemiLattice <: SemiRing <: KleeneAlg <: ...
M(SL) M(SR) M(KA) ...

Typage, suite

- ▶ Avantages :
 - ▶ on passe aux matrices :
“les algèbres de Kleene typées sont stables par formation de matrices quelconques” ;
 - ▶ on pourra considérer des relations binaires hétérogènes ;
 - ▶ ça frime, c'est des catégories.
- ▶ Inconvénients :
 - ▶ énoncés légèrement alourdis ;

Typage, suite

- ▶ Avantages :
 - ▶ on passe aux matrices :
“les algèbres de Kleene typées sont stables par formation de matrices quelconques” ;
 - ▶ on pourra considérer des relations binaires hétérogènes ;
 - ▶ ça frime, c'est des catégories.
- ▶ Inconvénients :
 - ▶ énoncés légèrement alourdis ;
- ▶ Nouvelle question :

$$a \cdot (b \cdot a)^* = (a \cdot b)^* \cdot a$$

$$a : A \rightarrow B$$

$$b : B \rightarrow A$$

Comment étendre les procédures de décision à ces algèbres typées ?

Modèles syntaxiques - cas non typé

- ▶ On veut prouver des égalités dans une algèbre quelconque (abstraite) \mathcal{A} , de signature T .
- ▶ Pour calculer, réduire des expressions, construire des automates, etc. . . on a besoin de passer dans un modèle “syntaxique” : $T\Sigma$ (Σ étant un ensemble “d’atomes”).
- ▶ Il suffit pour cela de trouver un homomorphisme $f : T\Sigma \rightarrow \mathcal{A}$

$$\frac{\frac{T\Sigma \vdash u = v}{\mathcal{A} \vdash f(u) = f(v)} \text{Morphism}}{\mathcal{A} \vdash \alpha = \beta} \text{Quote}$$

Modèles syntaxiques - cas non typé

$$\frac{\mathcal{T}\Sigma \vdash u = v}{\mathcal{A} \vdash f(u) = f(v)} \text{Morphism}$$
$$\frac{\mathcal{A} \vdash f(u) = f(v)}{\mathcal{A} \vdash \alpha = \beta} \text{Quote}$$

$$\mathcal{A} \vdash a \cdot b = a \cdot 1 \cdot b$$

Modèles syntaxiques - cas non typé

$$\frac{\mathcal{T}\Sigma \vdash u = v}{\mathcal{A} \vdash f(u) = f(v)} \text{Morphism}$$
$$\frac{\mathcal{A} \vdash f(u) = f(v)}{\mathcal{A} \vdash \alpha = \beta} \text{Quote}$$

$$\frac{\mathcal{A} \vdash f(d \ x_1 \ x_2) = f(d \ x_1 \ (d \ o \ x_2))}{\mathcal{A} \vdash a \cdot b = a \cdot 1 \cdot b} \text{Quote}$$

$\Sigma = \text{nat}$

$g : 1 \mapsto a, 2 \mapsto b$

$f = \widehat{g} : \mathcal{T}\Sigma \rightarrow \mathcal{A}$

Modèles syntaxiques - cas non typé

$$\frac{\frac{T\Sigma \vdash u = v}{\mathcal{A} \vdash f(u) = f(v)} \text{Morphism}}{\mathcal{A} \vdash \alpha = \beta} \text{Quote}$$

$$\frac{\frac{\frac{}{T\Sigma \vdash d \ x_1 \ x_2 \equiv d \ x_1 \ (d \ o \ x_2)} \text{Morphism}}{\mathcal{A} \vdash f(d \ x_1 \ x_2) = f(d \ x_1 \ (d \ o \ x_2))} \text{Quote}}{\mathcal{A} \vdash a \cdot b = a \cdot 1 \cdot b}$$

$\Sigma = \text{nat}$

$g : 1 \mapsto a, 2 \mapsto b$

$f = \widehat{g} : T\Sigma \rightarrow \mathcal{A}$

Modèles syntaxiques - cas non typé

$$\frac{\mathcal{T}\Sigma \vdash u = v}{\mathcal{A} \vdash f(u) = f(v)} \text{Morphism}$$
$$\frac{\mathcal{A} \vdash f(u) = f(v)}{\mathcal{A} \vdash \alpha = \beta} \text{Quote}$$

$$\frac{\frac{\frac{\text{bool} \vdash \text{dec}(d\ x_1\ x_2)(d\ x_1\ (d\ o\ x_2)) = \text{true}}{\mathcal{T}\Sigma \vdash d\ x_1\ x_2 \equiv d\ x_1\ (d\ o\ x_2)}}{\mathcal{A} \vdash f(d\ x_1\ x_2) = f(d\ x_1\ (d\ o\ x_2))} \text{Morphism}}{\mathcal{A} \vdash a \cdot b = a \cdot 1 \cdot b} \text{Quote}$$

$\Sigma = \text{nat}$

$g : 1 \mapsto a, 2 \mapsto b$

$f = \widehat{g} : \mathcal{T}\Sigma \rightarrow \mathcal{A}$

Modèles syntaxiques - cas non typé

$$\frac{\frac{T\Sigma \vdash u = v}{\mathcal{A} \vdash f(u) = f(v)} \text{Morphism}}{\mathcal{A} \vdash \alpha = \beta} \text{Quote}$$

$$\frac{\frac{\frac{\frac{bool \vdash true = true}{bool \vdash dec(d\ x_1\ x_2)(d\ x_1\ (d\ o\ x_2)) = true}}{T\Sigma \vdash d\ x_1\ x_2 \equiv d\ x_1\ (d\ o\ x_2)} \text{Morphism}}{\mathcal{A} \vdash f(d\ x_1\ x_2) = f(d\ x_1\ (d\ o\ x_2))} \text{Quote}}{\mathcal{A} \vdash a \cdot b = a \cdot 1 \cdot b} \text{Quote}$$

$\Sigma = nat$

$g : 1 \mapsto a, 2 \mapsto b$

$f = \widehat{g} : T\Sigma \rightarrow \mathcal{A}$

Modèles syntaxiques - cas typé

- ▶ Première idée : il suffit d'utiliser des foncteurs plutôt que des homomorphismes.
 - ▶ Aucun problème théorique,
 - ▶ mais la fonctionnalité “quote” de Coq ne permet pas de trouver automatiquement ces foncteurs.

Modèles syntaxiques - cas typé

- ▶ Première idée : il suffit d'utiliser des foncteurs plutôt que des homomorphismes.
 - ▶ Aucun problème théorique,
 - ▶ mais la fonctionnalité “quote” de Coq ne permet pas de trouver automatiquement ces foncteurs.
- ▶ Seconde idée : profitons-en pour **effacer les types** !

$$\frac{\frac{T\Sigma \vdash u = v}{\mathcal{A} \vdash f_{\Gamma, A, B}(u) = f_{\Gamma, A, B}(v) : A \rightarrow B}}{\mathcal{A} \vdash \alpha = \beta : A \rightarrow B}$$

Modèles syntaxiques - cas typé

- ▶ Première idée : il suffit d'utiliser des foncteurs plutôt que des homomorphismes.
 - ▶ Aucun problème théorique,
 - ▶ mais la fonctionnalité “quote” de Coq ne permet pas de trouver automatiquement ces foncteurs.
- ▶ Seconde idée : profitons-en pour **effacer les types** !

$$\frac{\frac{T\Sigma \vdash u = v}{\mathcal{A} \vdash f_{\Gamma, A, B}(u) = f_{\Gamma, A, B}(v) : A \rightarrow B}}{\mathcal{A} \vdash \alpha = \beta : A \rightarrow B}$$

Problèmes :

- ▶ f est nécessairement partielle : il existe généralement des éléments syntaxiques non typables, donc non évaluables ;
- ▶ f n'est pas fonctionnelle : certains éléments admettent plusieurs types, voire plusieurs évaluations, même à typage fixé.

Évaluation relationnelle

On va plutôt utiliser un **prédicat d'évaluation et de typage** :

$$\Gamma \vdash u \triangleright \alpha : A \rightarrow B$$

i.e., “dans le contexte Γ , u peut s'évaluer en α , de type $A \rightarrow B$ ”.

Évaluation relationnelle

On va plutôt utiliser un **prédicat d'évaluation et de typage** :

$$\Gamma \vdash u \triangleright \alpha : A \rightarrow B$$

i.e., “dans le contexte Γ , u peut s'évaluer en α , de type $A \rightarrow B$ ”.

$$u, v ::= o \mid z \mid x_i \mid d \ u \ v \mid p \ u \ v \mid \dots \quad (\text{T})$$

$$\frac{}{\Gamma; (i, \alpha : A \rightarrow B) \vdash x_i \triangleright \alpha : A \rightarrow B}$$

$$\frac{}{\Gamma \vdash o \triangleright 1 : A \rightarrow A}$$

$$\frac{\Gamma \vdash u \triangleright \alpha : A \rightarrow B \quad \Gamma \vdash v \triangleright \beta : B \rightarrow C}{\Gamma \vdash d \ u \ v \triangleright \alpha \cdot \beta : A \rightarrow C}$$

$$\frac{}{\Gamma \vdash z \triangleright 0 : A \rightarrow B}$$

$$\frac{\Gamma \vdash u \triangleright \alpha : A \rightarrow B \quad \Gamma \vdash v \triangleright \beta : A \rightarrow B}{\Gamma \vdash p \ u \ v \triangleright \alpha + \beta : A \rightarrow B}$$

...

Validité de l'effaçage des types

- Pour passer dans le modèle syntaxique en effaçant les types, il faut que le théorème suivant soit valide :

$$\frac{T\Sigma \vdash u = v \quad \Gamma \vdash u \triangleright \alpha : A \rightarrow B \quad \Gamma \vdash v \triangleright \beta : A \rightarrow B}{\mathcal{A} \vdash \alpha = \beta : A \rightarrow B} ?$$

Validité de l'effaçage des types

- ▶ Pour passer dans le modèle syntaxique en effaçant les types, il faut que le théorème suivant soit valide :

$$\frac{T\Sigma \vdash u = v \quad \Gamma \vdash u \triangleright \alpha : A \rightarrow B \quad \Gamma \vdash v \triangleright \beta : A \rightarrow B}{\mathcal{A} \vdash \alpha = \beta : A \rightarrow B} ?$$

- ▶ Cela dépend bien sûr de la théorie considérée :
 - ▶ pour les treillis, c'est trivial ;
 - ▶ pour les monoïdes, c'est relativement facile ;
 - ▶ pour les semi-anneaux et les algèbres de Kleene, c'est plus délicat, mais c'est vrai ;
 - ▶ pour les semi-anneaux résiduels, il y a des contraintes...

Quote (typé) en Ltac

- ▶ Ce détour par les types nous a permis de réaliser qu'on pouvait implémenter “quote” très simplement, dans Ltac :

$$\frac{T\Sigma \vdash u = v \quad \Gamma \vdash u \triangleright \alpha : A \rightarrow B \quad \Gamma \vdash v \triangleright \beta : A \rightarrow B}{\mathcal{A} \vdash \alpha = \beta : A \rightarrow B}$$

- ▶ Il suffit d'utiliser `eapply`, couplé à une représentation adéquate pour les environnements : démo.

Plan

Motivations

Décision des algèbres de Kleene par les automates

Rappel de la méthode

Vision algébrique de l'algorithme

Quelques détails d'implémentation Coq

Couches sous-jacentes du développement

Algèbres, matrices

Typage

Effaçage des types

Conclusions et perspectives

Conclusions

- ▶ Une tactique de décision des algèbres de Kleene :
 - ▶ par réflexion,
 - ▶ relativement efficace sur des exemples concrets.

Conclusions

- ▶ Une tactique de décision des algèbres de Kleene :
 - ▶ par réflexion,
 - ▶ relativement efficace sur des exemples concrets.

- ▶ D'autres outils pour les structures sous-jacentes :
 - ▶ algèbres diverses,
 - ▶ matrices,
 - ▶ beaucoup de réflexion sur la structure (modules vs typeclasses).

Conclusions

- ▶ Une tactique de décision des algèbres de Kleene :
 - ▶ par réflexion,
 - ▶ relativement efficace sur des exemples concrets.
- ▶ D'autres outils pour les structures sous-jacentes :
 - ▶ algèbres diverses,
 - ▶ matrices,
 - ▶ beaucoup de réflexion sur la structure (modules vs typeclasses).
- ▶ Typage (catégories) pour pouvoir travailler avec des matrices ;
 - ▶ théorèmes d'effaçage des types,
 - ▶ quote typé, en Ltac.

Perspectives

- ▶ Fin des preuves et peaufinage.
- ▶ Optimisations :
 - ▶ entiers binaires,
 - ▶ autres algorithmes de construction.

Perspectives

- ▶ Fin des preuves et peaufinage.
- ▶ Optimisations :
 - ▶ entiers binaires,
 - ▶ autres algorithmes de construction.
- ▶ Au delà des algèbres de Kleene ?
 - ▶ Converse (pas d'interaction)
 - ▶ Semi-treillis résiduels (système de Gentzen)
 - ▶ Algèbres/Treillis d'actions (réécriture, décidable?)
 - ▶ Allégories, algèbres de relations (indécidable)

Perspectives

- ▶ Fin des preuves et peaufinage.
- ▶ Optimisations :
 - ▶ entiers binaires,
 - ▶ autres algorithmes de construction.
- ▶ Au delà des algèbres de Kleene ?
 - ▶ Converse (pas d'interaction)
 - ▶ Semi-treillis résiduels (système de Gentzen)
 - ▶ Algèbres/Treillis d'actions (réécriture, décidable?)
 - ▶ Allégories, algèbres de relations (indécidable)
- ▶ Logique de Hoare, KAT.

Perspectives

- ▶ Fin des preuves et peaufinage.
- ▶ Optimisations :
 - ▶ entiers binaires,
 - ▶ autres algorithmes de construction.
- ▶ Au delà des algèbres de Kleene ?
 - ▶ Converse (pas d'interaction)
 - ▶ Semi-treillis résiduels (système de Gentzen)
 - ▶ Algèbres/Treillis d'actions (réécriture, décidable?)
 - ▶ Allégories, algèbres de relations (indécidable)
- ▶ Logique de Hoare, KAT.

Merci !

Nos questions

- ▶ Typeclasses vs. Structures canoniques vs. Modules
- ▶ Types “fantômes”
- ▶ Réécriture modulo A ou AC
- ▶ Induction bien fondée