

A Reflexive Tactic for Deciding Kleene Algebras

Thomas Braibant et Damien Pous (Grenoble)

1st Coq Workshop

Motivations

- ▶ Ease the formalisation of proofs dealing with **binary relations** in Coq (bisimulations...)

Motivations

- ▶ Ease the formalisation of proofs dealing with **binary relations** in Coq (bisimulations...)
- ▶ [Tarski et al.] : no finite axiomatisation
- ▶ A lot of partial axiomatisations
 - ▶ non-commutative monoids $(\cdot, 1)$
 - ▶ semi-lattices $(+, 0)$
 - ▶ non-commutative idempotent semirings $(\cdot, +, 1, 0)$
 - ▶ Kleene algebras $(\cdot, +, \star, 1, 0)$
 - ▶ Residuated semi-lattices $(\cdot, +, /, \backslash, 1, 0)$
 - ▶ Action algebras (Pratt) $(\cdot, +, /, \backslash, \star, 1, 0)$
 - ▶ Allegories (Freyd & Scedrov) $(\cdot, +, \wedge, /, \backslash, \bar{\cdot}, 1, 0)$
- ▶ In each case, different decidability / complexity properties

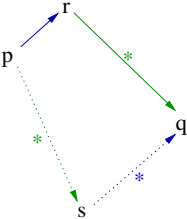
Motivations

- ▶ Ease the formalisation of proofs dealing with **binary relations** in Coq (bisimulations...)
- ▶ [Tarski et al.] : no finite axiomatisation
- ▶ A lot of partial axiomatisations
 - ▶ non-commutative monoids $(\cdot, 1)$
 - ▶ semi-lattices $(+, 0)$
 - ▶ non-commutative idempotent semirings $(\cdot, +, 1, 0)$
 - ▶ **Kleene algebras** $(\cdot, +, *, 1, 0)$
 - ▶ Residuated semi-lattices $(\cdot, +, /, \backslash, 1, 0)$
 - ▶ Action algebras (Pratt) $(\cdot, +, /, \backslash, *, 1, 0)$
 - ▶ Allegories (Freyd & Scedrov) $(\cdot, +, \wedge, /, \backslash, \bar{\cdot}, 1, 0)$
- ▶ In each case, different decidability / complexity properties

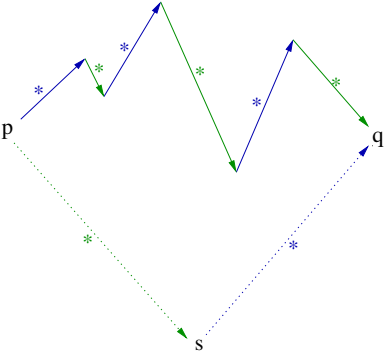
Motivations

- ▶ Ease the formalisation of proofs dealing with **binary relations** in Coq (bisimulations...)
- ▶ [Tarski et al.] : no finite axiomatisation
- ▶ A lot of partial axiomatisations
 - ▶ non-commutative monoids $(\cdot, 1)$
 - ▶ semi-lattices $(+, 0)$
 - ▶ non-commutative idempotent semirings $(\cdot, +, 1, 0)$
 - ▶ **Kleene algebras** $(\cdot, +, *, 1, 0)$
 - ▶ Residuated semi-lattices $(\cdot, +, /, \backslash, 1, 0)$
 - ▶ Action algebras (Pratt) $(\cdot, +, /, \backslash, *, 1, 0)$
 - ▶ Allegories (Freyd & Scedrov) $(\cdot, +, \wedge, /, \backslash, \bar{\cdot}, 1, 0)$
- ▶ In each case, different decidability / complexity properties
- ▶ Example : “Weak confluence implies the Church-Rosser property”

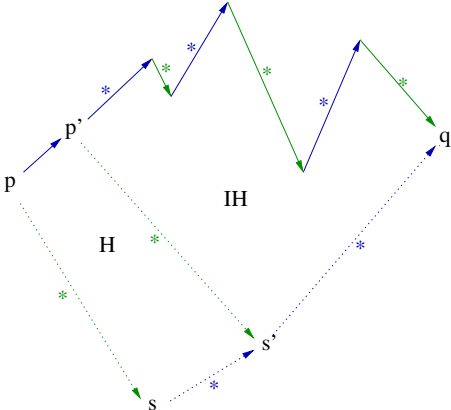
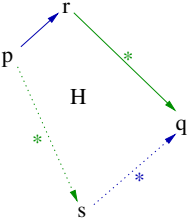
Church-Rosser



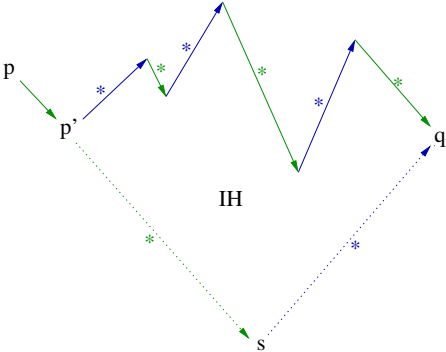
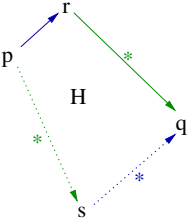
implies



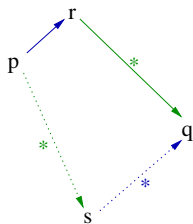
Church-Rosser (Diagrammatic proof)



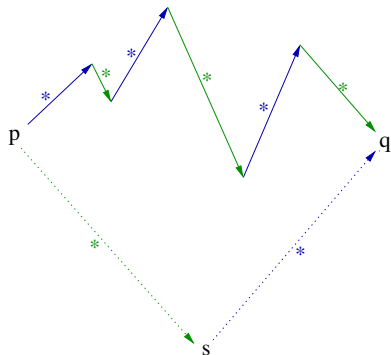
Church-Rosser (Diagrammatic proof)



Church-Rosser (more formally)



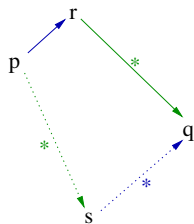
implies



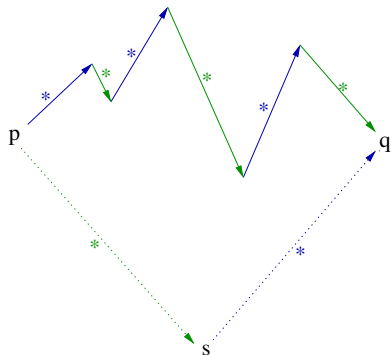
$$(\forall p, r, q, pRr, rS^*q \Rightarrow \exists s, pS^*s \wedge sR^*q)$$

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

Church-Rosser (more formally)



implies



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

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

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

churchrosser.v

Objectives

- ▶ The algebraic view improves :
 - ▶ goals readability ;
- ▶ but we saw the need for :
 - ▶ decision tactics (à la ring, omega) :
`kleene_reflexivity, monoid_reflexivity, semiring_reflexivity...`
 - ▶ simplification tactics (ring_simplify) :
`semiring_normalize, aci_normalize...`
 - ▶ rewriting tactics (modulo A, modulo AC) :
`monoid_rewrite, ac_rewrite.`

Outline

Motivations

Deciding Kleene Algebras in Coq

Underlying parts of the development

Conclusions and perspectives

Scott vs. Kozen

Let α and β be two regular expressions ($+$, \cdot , 0 , 1 , $*$).

Scott '50 α and β represent the same language iff the corresponding minimal automata are isomorphic.

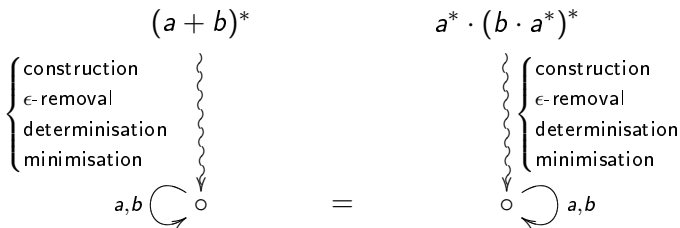
Scott vs. Kozen

Let α and β be two regular expressions ($+$, \cdot , 0 , 1 , $*$).

Scott '50 α and β represent the same language iff the corresponding minimal automata are isomorphic.

Kozen '94 Initiality of this model for Kleene algebras :
If α and β lead to the same automata,
then $\mathcal{A} \vdash \alpha = \beta$, for any Kleene algebra \mathcal{A} .

Scott vs. Kozen (again)



Scott '50 : We deduce $L((a + b)^*) = L(a^* \cdot (b \cdot a^*)^*)$.

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

Making a reflexive tactic

- ▶ Theoretical complexity is exponential...
 - ▶ it's tractable in practice...
 - ▶ as long as we take some care in the implementation

Making a reflexive tactic

- ▶ Theoretical complexity is exponential...
 - ▶ it's tractable in practice...
 - ▶ as long as we take some care in the implementation
- ▶ Coq is a programming language, we code the algorithm :
`Definition decide_Kleene : regexp → regexp → bool := ...`

Making a reflexive tactic

- ▶ Theoretical complexity is exponential...
 - ▶ it's tractable in practice...
 - ▶ as long as we take some care in the implementation
- ▶ Coq is a programming language, we code the algorithm :
`Definition decide_Kleene : regexp → regexp → bool := ...`
- ▶ We formalize Kozen's proof in Coq :
`Theorem Kozen : ∀ a b : regexp, decide_Kleene a b = true → a == b.`
(`==` is the equality generated by the axioms of Kleene Algebras)
- ▶ Then we wrap this in a tactic.

Kozen's Proof

- ▶ The main idea is to represent automata algebraically, with matrices :

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

Kozen's Proof

- ▶ The main idea is to represent automata algebraically, with matrices :

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

- ▶ Matrices over a Kleene algebra form a Kleene algebra.

Kozen's Proof

- ▶ The main idea is to represent automata algebraically, with matrices :

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

- ▶ Matrices over a Kleene algebra form a Kleene algebra.

Kozen's Proof

- ▶ The main idea is to represent automata algebraically, with matrices :

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

- ▶ Matrices over a Kleene algebra form a Kleene algebra.
- ▶ Transcribe and validate the algorithms in this algebraic setting.

Kozen's Proof

- ▶ The main idea is to represent automata algebraically, with matrices :

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

- ▶ Matrices over a Kleene algebra form a Kleene algebra.
- ▶ Transcribe and validate the algorithms in this algebraic setting.
in this talk, only a glimpse of these

Thompson's construction (Dot)

Construction of the automaton of $\alpha \cdot \beta$

▶ If $\alpha = u \cdot M^* \cdot v$ and $\beta = s \cdot N^* \cdot t$

▶ Then

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

Thompson's construction (Dot)

Construction of the automaton of $\alpha \cdot \beta$

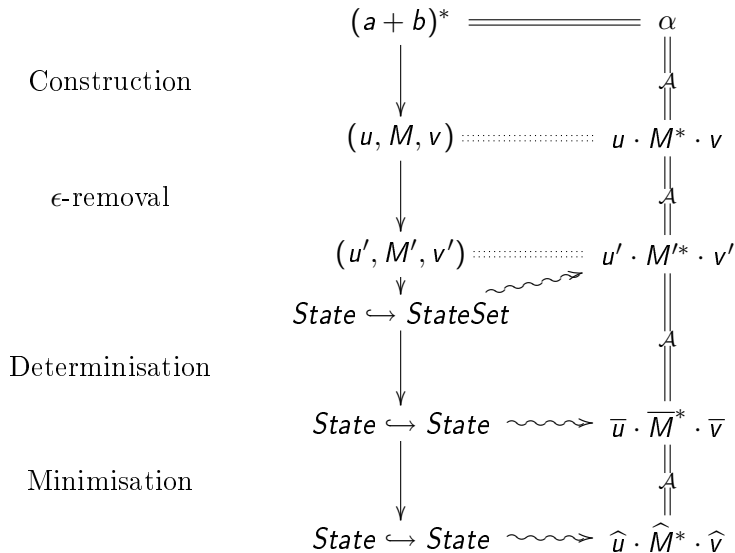
▶ If $\alpha = u \cdot M^* \cdot v$ and $\beta = s \cdot N^* \cdot t$

▶ Then

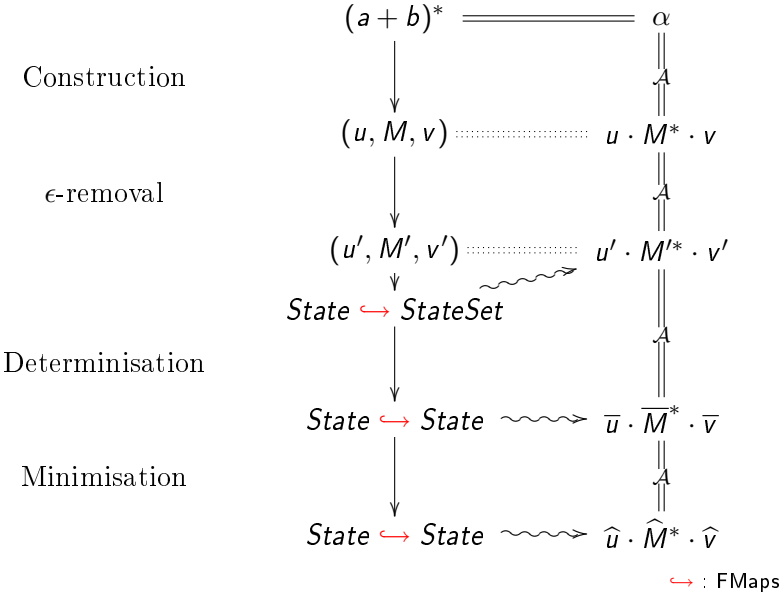
$$\begin{aligned} & \left[u \mid 0 \right] \cdot \left[\begin{array}{c|c} M & v \cdot s \\ \hline 0 & N \end{array} \right]^* \cdot \left[\begin{array}{c} 0 \\ t \end{array} \right] \\ &= \left[u \mid 0 \right] \cdot \left[\begin{array}{c|c} M^* & M^* \cdot v \cdot s \cdot N^* \\ \hline 0 & N^* \end{array}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

other cases are similar

Datastructures (FSets/FMaps)



Datastructures (FSets/FMaps)



Plan

Motivations

Deciding Kleene Algebras in Coq

Underlying parts of the development

Conclusions and perspectives

Algebraic hierarchy

- ▶ We want to share the tools we developed for monoids, lattices, semi-rings, etc...
(e.g., `semiring_reflexivity` in the context of a Kleene algebra.)

- ▶ We follow the mathematical algebraic hierarchy using **Typeclasses**

SemiLattice

<: SemiRing <: KleeneAlg <: ...

Monoid

Matrices

- ▶ Infinite functions, with a constrained pointwise equality :

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

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

Matrices

- ▶ Infinite functions, with a constrained pointwise equality :

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

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

- ▶ Easy to manipulate (proof/programs separation)

$$(M * N)_{i,j} = \left(\sum_{k=0}^m M_{i,k} * N_{k,j} \right)_{i,j}$$

Fixpoint $\text{sum } i \ k \ (f : \text{nat} \rightarrow X) :=$
match k **with** $0 \Rightarrow 0 \mid S \ k \Rightarrow f \ i + \text{sum } (S \ i) \ k \ f$ **end**.

Definition $\text{dot } n \ m \ p \ (M : \text{MX } n \ m) \ (N : \text{MX } m \ p) :=$
 $\text{fun } i \ j \Rightarrow \text{sum } 0 \ m \ (\text{fun } k \Rightarrow M \ i \ k * N \ k \ j).$

Matrices

- ▶ Infinite functions, with a constrained pointwise equality :

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

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

- ▶ Easy to manipulate (proof/programs separation)

$$(M * N)_{i,j} = \left(\sum_{k=0}^m M_{i,k} * N_{k,j} \right)_{i,j}$$

Fixpoint $\text{sum } i \ k \ (f : \text{nat} \rightarrow X) :=$
match k **with** $0 \Rightarrow 0 \mid S \ k \Rightarrow f \ i + \text{sum } (S \ i) \ k \ f$ **end.**

Definition $\text{dot } n \ m \ p \ (M : \text{MX } n \ m) \ (N : \text{MX } m \ p) :=$
 $\text{fun } i \ j \Rightarrow \text{sum } 0 \ m \ (\text{fun } k \Rightarrow M \ i \ k * N \ k \ j).$

bounds proofs are easy to cope with
no hidden boilerplate

Matrices cont.

Thanks to typeclasses, we inherit tools and theorems for matrices :

- ▶ square matrices built over a semi-ring form a semi-ring;
- ▶ square matrices built over a Kleene algebra form a Kleene algebra.

`matrices.v`

How to deal with rectangular matrices ?

- ▶ Without extra stuff, we cannot re-use tools for them : rectangular matrices do not form a semiring
 - ▶ operations $(\cdot, +, \dots)$ are partial (dimensions have to agree)

`X`: Type.

`dot` : $X \rightarrow X \rightarrow X$.

`one` : X .

`plus` : $X \rightarrow X \rightarrow X$.

`zero` : X .

`star` : $X \rightarrow X$.

`dot_neutral_left` :

$\forall x, \text{dot one } x = x$.

...

How to deal with rectangular matrices ?

- ▶ Without extra stuff, we cannot re-use tools for them : rectangular matrices do not form a semiring
 - ▶ operations $(\cdot, +, \dots)$ are partial (dimensions have to agree)

X : Type.	T : Type. X : $T \rightarrow T \rightarrow \text{Type}$.
dot : $X \rightarrow X \rightarrow X$. one : X .	dot : $\forall n\ m\ p, X\ n\ m \rightarrow X\ m\ p \rightarrow X\ n\ p$. one : $\forall n, X\ n\ n$.
plus : $X \rightarrow X \rightarrow X$. zero : X .	plus : $\forall n\ m, X\ n\ m \rightarrow X\ n\ m \rightarrow X\ n\ m$. zero : $\forall n\ m, X\ n\ m$.
star : $X \rightarrow X$.	star : $\forall n, X\ n\ n \rightarrow X\ n\ n$.
dot_neutral_left : $\forall x, \text{dot one } x = x$.	dot_neutral_left : $\forall n\ m (x: X\ n\ m), \text{dot one } x = x$.
...	...

How to deal with rectangular matrices ?

- ▶ Without extra stuff, we cannot re-use tools for them : rectangular matrices do not form a semiring
 - ▶ operations $(\cdot, +, \dots)$ are partial (dimensions have to agree)

$X: \text{Type}.$	$T: \text{Type}.$
	$X: T \rightarrow T \rightarrow \text{Type}.$
$\text{dot}: X \rightarrow X \rightarrow X.$	$\text{dot}: \forall n\ m\ p, X\ n\ m \rightarrow X\ m\ p \rightarrow X\ n\ p.$
$\text{one}: X.$	$\text{one}: \forall n, X\ n\ n.$
$\text{plus}: X \rightarrow X \rightarrow X.$	$\text{plus}: \forall n\ m, X\ n\ m \rightarrow X\ n\ m \rightarrow X\ n\ m.$
$\text{zero}: X.$	$\text{zero}: \forall n\ m, X\ n\ m.$
$\text{star}: X \rightarrow X.$	$\text{star}: \forall n, X\ n\ n \rightarrow X\ n\ n.$
$\text{dot_neutral_left}: \forall x, \text{dot_one}\ x = x.$	$\text{dot_neutral_left}: \forall n\ m\ (x: X\ n\ m), \text{dot_one}\ x = x.$
...	...

- ▶ Introduce **typed** (graded) structures from the beginning

Typed structures

We handle heterogeneous relations ($X A B := A \rightarrow B \rightarrow \text{Prop}$),
as well as matrices :

```
MxSemiLattice    : SemiLattice    → SemiLattice .  
MxSemiRing      : SemiRing        → SemiRing .  
MxKleeneAlgebra : KleeneAlgebra → KleeneAlgebra .
```

Here, we deal with typed structures

- ▶ All theorems are inherited at the matricial level.

Typed structures

We handle heterogeneous relations ($X A B := A \rightarrow B \rightarrow \text{Prop}$),
as well as matrices :

```
MxSemiLattice    : SemiLattice    → SemiLattice .  
MxSemiRing      : SemiRing        → SemiRing .  
MxKleeneAlgebra : KleeneAlgebra  → KleeneAlgebra .
```

Here, we deal with typed structures

- ▶ All theorems are inherited at the matricial level.
- ▶ What about extending decision procedures to deal with typed structures?

$$\begin{array}{ccc} a \cdot (b \cdot a)^* & \stackrel{?}{=} & (a \cdot b)^* \cdot a \\ \downarrow & & \downarrow \\ \bullet & = & \bullet \end{array}$$

Typed structures

We handle heterogeneous relations ($X A B := A \rightarrow B \rightarrow \text{Prop}$),
as well as matrices :

```
MxSemiLattice    : SemiLattice    → SemiLattice .  
MxSemiRing      : SemiRing        → SemiRing .  
MxKleeneAlgebra : KleeneAlgebra  → KleeneAlgebra .
```

Here, we deal with typed structures

- ▶ All theorems are inherited at the matricial level.
- ▶ What about extending decision procedures to deal with typed structures?

$$\begin{array}{ccc} a \cdot (b \cdot a)^* & \stackrel{?}{=} & (a \cdot b)^* \cdot a \\ \downarrow & & \downarrow \\ \bullet & = & \bullet \end{array}$$


Typed structures

We handle heterogeneous relations ($X \ A \ B := A \rightarrow B \rightarrow \text{Prop}$),
as well as matrices :

```
MxSemiLattice    : SemiLattice    → SemiLattice .  
MxSemiRing      : SemiRing       → SemiRing .  
MxKleeneAlgebra : KleeneAlgebra  → KleeneAlgebra .
```

Here, we deal with typed structures

- ▶ All theorems are inherited at the matricial level.
- ▶ What about extending decision procedures to deal with typed structures?

$$a \cdot (b \cdot a)^* \quad \stackrel{?}{=} \quad (a \cdot b)^* \cdot a \quad : \quad A \rightarrow B$$


The diagram shows two expressions, $a \cdot (b \cdot a)^*$ on the left and $(a \cdot b)^* \cdot a$ on the right, separated by an equals sign with a question mark above it. Below each expression is a wavy arrow pointing down to a solid black dot.

$$a : A \rightarrow B, b : B \rightarrow A$$


Typed structures

We handle heterogeneous relations ($X A B := A \rightarrow B \rightarrow \text{Prop}$),
as well as matrices :

```
MxSemiLattice    : SemiLattice    → SemiLattice .  
MxSemiRing      : SemiRing        → SemiRing .  
MxKleeneAlgebra : KleeneAlgebra  → KleeneAlgebra .
```

Here, we deal with typed structures

- ▶ All theorems are inherited at the matricial level.
- ▶ What about extending decision procedures to deal with typed structures?

$$a \cdot (b \cdot a)^* \quad \stackrel{?}{=} \quad (a \cdot b)^* \cdot a \quad : \quad A \rightarrow B$$


The diagram shows the equation $a \cdot (b \cdot a)^* \stackrel{?}{=} (a \cdot b)^* \cdot a : A \rightarrow B$. Below the first expression $a \cdot (b \cdot a)^*$ and the second expression $(a \cdot b)^* \cdot a$, there are wavy lines that converge to a solid black dot. A horizontal equals sign (=) is positioned between these two dots, indicating that the two expressions are being compared or equated.

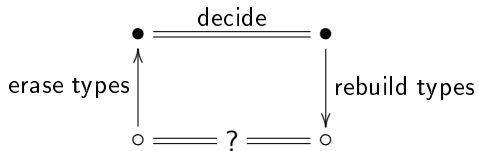
$a : A \rightarrow B, b : B \rightarrow A$

tackle the problem differently... let's erase types!

Untyping

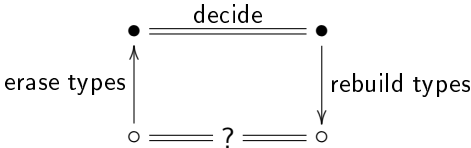
untyped setting :

typed setting :



Untyping

untyped setting :



typed setting :

► Depending on the algebraic setting :

\mathcal{A}	
semi-lattices	trivial
monoids	rather easy
semirings	tricky
Kleene algebras	same as for semirings
residuated lattices	with constraints
action algebras/lattices	?

This result is crucial for this work...

Plan

Motivations

Deciding Kleene Algebras in Coq

Underlying parts of the development

Conclusions and perspectives

Conclusions

- ▶ A decision tactic for Kleene algebras (available on the web) :
 - ▶ reflexive
 - ▶ (relatively) efficient
 - ▶ 234 Kb of compressed .v files using gzip (current trunk)
 - ▶ 1.417 Pous/year

Conclusions

- ▶ A decision tactic for Kleene algebras (available on the web) :
 - ▶ reflexive
 - ▶ (relatively) efficient
 - ▶ 234 Kb of compressed .v files using gzip (current trunk)
 - ▶ 1.417 Pous/year

- ▶ Other tools for the underlying structures :
 - ▶ various (graded) algebraic settings,
 - ▶ matrices,

Learnings

- ▶ FSets/FMaps :
 - ▶ used a lot in our development
 - ▶ instantiation time of multiple applications of functors (sets of sets of ...) can become huge
 - ▶ design choice : underlying ordered types mixing proofs and programs
- ▶ Typeclasses :
 - ▶ much more supple to use than modules
 - ▶ there is an overhead due to the inference of implicit arguments
- ▶ JMeq :
 - ▶ the axiom is necessary to prove the untyping lemmas

What's coming next

- ▶ More efficient tactic
 - ▶ binary integers (positive and FMapPositive)
 - ▶ better construction algorithm
 - ▶ Spaghetti Hopcroft
- ▶ release of our Matrices library :
 - + ring like tactics for rectangular matrices

Questions ?

Determinisation

- ▶ Construct the powerset automata
- ▶ Let X be the decoding matrix of the accessible subsets of the automata (u, M, v) :

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

- ▶ We can define \overline{M} and \overline{u} such that :

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

- ▶ We deduce

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