Trusted Extension in Coq

Laurent Théry

INRIA Sophia Antipolis France

Motivations

Motivations

Why trust is so important?

Motivations

Why trust is so important?

Mainly Proving Truth

Motivations

Why trust is so important?

- Mainly Proving Truth
- Keep the design simple

Motivations

Why trust is so important?

Mainly Proving Truth

Keep the design simple

$\longrightarrow$ Proving Trusted Base

Motivations

Why trust is so important?

Mainly Proving Truth

Keep the design simple

$\longrightarrow$ Proving Trusted Base

CCP (Code-Carrying Proofs)

Motivations

Why trust is so important?

- Mainly Proving Truth
- Keep the design simple
  - $\longrightarrow$ Proving Trusted Base
- CCP (Code-Carrying Proofs)
- Genericity $\rightarrow$ Domain Specific Applications

Outline

Outline

Overview of Coq

Outline

Overview of Coq

Reflection

Outline

Overview of Coq

Reflection

Extensions: Various Flavours

Outline

Overview of Coq

Reflection

Extensions: Various Flavours

More Extensions

Outline

Overview of Coq

Reflection

Extensions: Various Flavours

More Extensions

Conclusions

Coq

Coq

Logic: Calculus of Inductive Constructions

Coq

Logic: Calculus of Inductive Constructions

First version 86, now v8.2

http://coq.inria.fr/

Coq

Logic: Calculus of Inductive Constructions

First version 86, now v8.2

http://coq.inria.fr/

Trust in Coq

Coq

Logic: Calculus of Inductive Constructions

First version 86, now v8.2

http://coq.inria.fr/

Trust in Coq

Explicit Proof Object

Coq

Logic: Calculus of Inductive Constructions

First version 86, now v8.2

http://coq.inria.fr/

Trust in Coq

- Explicit Proof Object
- Trusted Component: coqchk

Coq

```
Inductive ℕ : Type := O | S (n:ℕ).
```

Coq

```
Inductive ℕ : Type := O | S (n:ℕ).

Inductive _ = _ : ∀T:Type, T → T → Prop :=
    refl_equal : ∀T:Type, ∀x:T, x = x.
```

Coq

```
Inductive ℕ : Type := O | S (n:ℕ).

Inductive _ = _ : ∀T:Type, T → T → Prop :=
    refl_equal : ∀T:Type, ∀x:T, x = x.

Fact triv : 2 = 2.
```

Coq

```
Inductive ℕ : Type := O | S (n:ℕ).

Inductive _ = _ : ∀T:Type, T → T → Prop :=
    refl_equal : ∀T:Type, ∀x:T, x = x.

Fact triv : 2 = 2.
Proof. apply refl_equal. Qed.
```

Coq

```
Inductive ℕ : Type := O | S (n:ℕ).

Inductive _ = _ : ∀T:Type, T → T → Prop :=
    refl_equal : ∀T:Type, ∀x:T, x = x.

Fact triv : 2 = 2.
Proof. apply refl_equal. Qed.

Print triv.
```

Coq

```
Inductive ℕ : Type := O | S (n:ℕ).

Inductive _ = _ : ∀T:Type, T → T → Prop :=
    refl_equal : ∀T:Type, ∀x:T, x = x.

Fact triv : 2 = 2.
Proof. apply refl_equal. Qed.

Print triv.
  triv = (refl_equal ℕ 2)
```

Coq

```
Function x + y :=
  if x is S x' then S(x' + y) else y.
```

Coq

```
Function x + y :=
  if x is S x' then S(x' + y) else y.

Fact triv' : 1 + 2 = 3.
```

Coq

```
Function x + y :=
  if x is S x' then S(x' + y) else y.

Fact triv' : 1 + 2 = 3.
Proof. apply refl_equal. Qed.
```

Coq

```
Function x + y :=
  if x is S x' then S(x' + y) else y.

Fact triv' : 1 + 2 = 3.
Proof. apply refl_equal. Qed.

Print triv'.
```

Coq

```
Function x + y :=
  if x is S x' then S(x' + y) else y.

Fact triv' : 1 + 2 = 3.
Proof. apply refl_equal. Qed.

Print triv'.
  triv' = (refl_equal ℕ 3)
```

Reflection

Reflection

Samuel Boutin ('97)

"Using reflection to build efficient and certified decision procedures"

Reflection

Samuel Boutin ('97)

"Using reflection to build efficient and certified decision procedures"

$$
\begin{aligned}
&(a^2+b^2+c^2+d^2+e^2+f^2+g^2+h^2)\,(m^2+n^2+o^2+p^2+q^2+r^2+s^2+t^2) = \\
&\quad (am-bn-co-dp-eq-fr-gs-ht)^2+ \\
&\quad (bm+an+do-cp+fq-er-hs+gt)^2+ \\
&\quad (cm-dn+ao+bp+gq+hr-es-ft)^2+ \\
&\quad (dm+cn-bo+ap+hq-gr+fs-et)^2+ \\
&\quad (em-fn-go-hp+aq+br+cs+dt)^2+ \\
&\quad (fm+en-ho+gp-bq+ar-ds+ct)^2+ \\
&\quad (gm+hn+eo-fp-cq+dr+as-bt)^2+ \\
&\quad (hm-gn+fo+ep-dq-cr+bs+at)^2
\end{aligned}
$$

Reflection

Reflection

Abelian Group:

```
Inductive expr: Type :=
 add (e1 e2: expr) | opp (e: expr) | var (n: ℕ) | zero.
```

Reflection

Abelian Group:

```
Inductive expr: Type :=
 add (e1 e2: expr) | opp (e: expr) | var (n: ℕ) | zero.
```

$(x + y) - y \rightsquigarrow$ `(add (add (var 1) (var 2)) (opp (var 1)))`

Reflection

Abelian Group:

```
Inductive expr: Type :=
 add (e1 e2: expr) | opp (e: expr) | var (n: ℕ) | zero.
```

List of integers

Reflection

Abelian Group:

```
Inductive expr: Type :=
 add (e1 e2: expr) | opp (e: expr) | var (n: ℕ) | zero.
```

List of integers

```
Function l1 ++ l2 :=
 if l1 is a :: l3 then
 if l2 is b :: l4 then

 else l1
 else l2
```

Reflection

Abelian Group:

```
Inductive expr: Type :=
 add (e1 e2: expr) | opp (e: expr) | var (n: ℕ) | zero.
```

List of integers

```
Function l1 ++ l2 :=
 if l1 is a :: l3 then
 if l2 is b :: l4 then
          if |a| < |b| then a :: (l3 ++ l2)
    else if |b| < |a| then b :: (l1 ++ l4)
    else if   a == b  then a :: b :: (l3 ++ l4)
    else (l3 ++ l4)
 else l1
 else l2
```

Reflection

Abelian Group:

```
Inductive expr: Type :=
 add (e1 e2: expr) | opp (e: expr) | var (n: ℕ) | zero.
```

List of integers

```
Function l1 ++ l2 :=
 if l1 is a :: l3 then
 if l2 is b :: l4 then
          if |a| < |b| then a :: (l3 ++ l2)
    else if |b| < |a| then b :: (l1 ++ l4)
    else if   a == b  then a :: b :: (l3 ++ l4)
    else (l3 ++ l4)
 else l1
 else l2
```

`[1,1,2] ++ [-1,2]` $\rightsquigarrow$ `[1,2,2]`

Reflection

Abelian Group:

```
Inductive expr: Type :=
 add (e1 e2: expr) | opp (e: expr) | var (n: ℕ) | zero.
```

List of integers

```
Function l1 ++ l2 :=
 if l1 is a :: l3 then
 if l2 is b :: l4 then
          if |a| < |b| then a :: (l3 ++ l2)
    else if |b| < |a| then b :: (l1 ++ l4)
    else if   a == b  then a :: b :: (l3 ++ l4)
    else (l3 ++ l4)
 else l1
 else l2
```

Reflection

Normalisation:

Reflection

Normalisation:

```
Function e2l e :=
 match e with
 | add e1 e2 ⇒ e2l e1 ++ e2l e2
 | opp e1    ⇒ map (fun x ⇒ −x) (e2l e1)
 | var n     ⇒ [+n]
 | zero      ⇒ []
 end
```

Reflection

Normalisation:

```
Function e2l e :=
 match e with
 | add e1 e2 ⇒ e2l e1 ++ e2l e2
 | opp e1    ⇒ map (fun x ⇒ −x) (e2l e1)
 | var n     ⇒ [+n]
 | zero      ⇒ []
 end

e2l (add (add (var 1) (var 2)) (opp (var 2)))
```

Reflection

Normalisation:

```
Function e2l e :=
 match e with
 | add e1 e2 ⇒ e2l e1 ++ e2l e2
 | opp e1    ⇒ map (fun x ⇒ −x) (e2l e1)
 | var n     ⇒ [+n]
 | zero      ⇒ []
 end
```

```
e2l (add (add (var 1) (var 2)) (opp (var 2))) ⇝
  ([1] ++ [2]) ++ (map (fun x ⇒ −x) [2])
```

Reflection

Normalisation:

```
Function e2l e :=
 match e with
 | add e1 e2 ⇒ e2l e1 ++ e2l e2
 | opp e1    ⇒ map (fun x ⇒ −x) (e2l e1)
 | var n     ⇒ [+n]
 | zero      ⇒ []
 end
```

```
e2l (add (add (var 1) (var 2)) (opp (var 2))) ⇝
  [1,2] ++ (map (fun x ⇒ −x) [2])
```

Reflection

Normalisation:

```
Function e2l e :=
 match e with
 | add e1 e2 ⇒ e2l e1 ++ e2l e2
 | opp e1    ⇒ map (fun x ⇒ −x) (e2l e1)
 | var n     ⇒ [+n]
 | zero      ⇒ []
 end
```

```
e2l (add (add (var 1) (var 2)) (opp (var 2))) ⇝
  [1,2] ++ [-2]
```

Reflection

Normalisation:

```
Function e2l e :=
 match e with
 | add e1 e2 ⇒ e2l e1 ++ e2l e2
 | opp e1    ⇒ map (fun x ⇒ −x) (e2l e1)
 | var n     ⇒ [+n]
 | zero      ⇒ []
 end

e2l (add (add (var 1) (var 2)) (opp (var 2))) ⇝
  [1]
```

Reflection

Interpretations:

Reflection

Interpretations:

```
Structure Group :=
    {
        T:  Type;
      add:  T -> T -> T;
      opp:  T -> T;
     zero:  T;
    assoc:  ∀x∀y∀z, add (add x y) = add x (add y z);
      com:  ∀x∀y, add x y = add y x;
    lefti:  ∀x, add x zero = x;
    lefto:  ∀x, add x (opp x) = zero;
    }
```

Reflection

Interpretations:

```
Structure Group :=
    {
        T:  Type;
      add:  T -> T -> T;
      opp:  T -> T;
     zero:  T;
    assoc:  ∀x∀y∀z, add (add x y) = add x (add y z);
      com:  ∀x∀y, add x y = add y x;
    lefti:  ∀x, add x zero = x;
    lefto:  ∀x, add x (opp x) = zero;
    }

Definition zGroup := mkGroup {ℤ,+,−,0,...}
```

Reflection

Interpretations:

```
Structure Group :=
    {
        T:  Type;
      add:  T -> T -> T;
      opp:  T -> T;
     zero:  T;
    assoc:  ∀x∀y∀z, add (add x y) = add x (add y z);
      com:  ∀x∀y, add x y = add y x;
    lefti:  ∀x, add x zero = x;
    lefto:  ∀x, add x (opp x) = zero;
    }

Definition zGroup := mkGroup {ℤ,+,−,0,...}

Definition oGroup := mkGroup {bool,⊕,¬,true,...}
```

Reflection

Interpretations:

Reflection

Interpretations:

```
Function [|e|]_{g,env} :=
 match e with
 | add e1 e2 ⇒ g.add [|e1|]_{g,env} [|e2|]_{g,env}
 | opp e1    ⇒ g.opp [|e1|]_{g,env}
 | var n     ⇒ env n
 | zero      ⇒ g.zero
 end
```

Reflection

Interpretations:

```
Function [|e|]_{g,env} :=
 match e with
 | add e1 e2 ⇒ g.add [|e1|]_{g,env} [|e2|]_{g,env}
 | opp e1    ⇒ g.opp [|e1|]_{g,env}
 | var n     ⇒ env n
 | zero      ⇒ g.zero
 end
```

$$[|\texttt{add (add (var 1) (var 2)) (opp (var 2))}|]_{\texttt{zGroup},\{1\mapsto x,2\mapsto y\}} \rightsquigarrow (x+y)+-y$$

Reflection

Interpretations:

```
Function [|e|]_{g,env} :=
 match e with
 | add e1 e2 ⇒ g.add [|e1|]_{g,env} [|e2|]_{g,env}
 | opp e1    ⇒ g.opp [|e1|]_{g,env}
 | var n     ⇒ env n
 | zero      ⇒ g.zero
 end
```

$$[|\texttt{add (add (var 1) (var 2)) (opp (var 2))}|]_{\texttt{zGroup},\{1\mapsto x,2\mapsto y\}} \rightsquigarrow (x+y)+-y$$

$$[|\texttt{add (add (var 1) (var 2)) (opp (var 2))}|]_{\texttt{oGroup},\{1\mapsto x,2\mapsto y\}} \rightsquigarrow (x \oplus y) \oplus \neg y$$

Reflection

Interpretations:

Reflection

Interpretations:

```
Definition  <|z|>_{g,env} :=
 if 0 < z then env z else g.opp (env −z)
```

Reflection

Interpretations:

```
Definition  <|z|>_{g,env} :=
 if 0 < z then env z else g.opp (env −z)

Function {|l|}_{g,env} :=
 match l with
 |  []    ⇒ g.zero
 |  [z]   ⇒ <|z|>_{g,env}
 |  z::l1 ⇒ g.add <|z|>_{g,env} {|l1|}_{g,env}
 end
```

Reflection

Interpretations:

```
Definition  <|z|>_{g,env} :=
 if 0 < z then env z else g.opp (env −z)

Function {|l|}_{g,env} :=
 match l with
 | []    ⇒ g.zero
 | [z]   ⇒ <|z|>_{g,env}
 | z::l1 ⇒ g.add <|z|>_{g,env} {|l1|}_{g,env}
 end
```

$\{|[1]|\}_{\texttt{zGroup},\{1\mapsto x,2\mapsto y\}} \rightsquigarrow x$

$\{|[1]|\}_{\texttt{oGroup},\{1\mapsto x,2\mapsto y\}} \rightsquigarrow x$

Reflection

Correctness:

Reflection

Correctness:

Lemma e2l_cor: $\forall g\, \forall env\, \forall e$, [|$e$|]$_{g,env}$ = {|e2l e|}$_{g,env}$.

Reflection

Correctness:

Lemma e2l_cor: $\forall g\, \forall env\, \forall e$, [|$e$|]$_{g,env}$ = {|e2l e|}$_{g,env}$.

Proof by structural induction on e

Reflection

Correctness:

Lemma e2l_cor: $\forall g\, \forall env\, \forall e,$ [|e|]$_{g,env}$ = {|e2l e|}$_{g,env}$.

Proof by structural induction on e

Proof of $\forall x \forall y,\ (x + y) + -y = x$:

Reflection

Correctness:

Lemma e2l_cor: $\forall g\, \forall env\, \forall e,$ [|e|]$_{g,env}$ = {|e2l e|}$_{g,env}$.

Proof by structural induction on e

Proof of $\forall x \forall y,\ (x + y) + -y = x$:

```
(fun x y ⇒
   (e2l_cor zGroup {1↦x,2↦y}
   (add (add (var 1) (var 2)) (opp (var 2)))))
```

Reflection

Correctness:

Lemma e2l_cor: $\forall g\, \forall env\, \forall e,$ [|e|]$_{g,env}$ = {|e2l e|}$_{g,env}$.

Proof by structural induction on e

Proof of $\forall x \forall y,\ (x + y) + -y = x$:

```
(fun x y ⇒
   (e2l_cor zGroup {1↦x,2↦y}
   (add (add (var 1) (var 2)) (opp (var 2)))))
```

Proof of $\forall x \forall y,\ (x \oplus y) \oplus \neg y = x$:

```
(fun x y ⇒
   (e2l_cor oGroup {1↦x,2↦y}
   (add (add (var 1) (var 2)) (opp (var 2)))))
```

Reflection

Correctness:

Lemma e2l_cor: $\forall g\, \forall env\, \forall e,$ [|e|]$_{g,env}$ = {|e2l e|}$_{g,env}$.

Proof by structural induction on e

Proof of $\forall x \forall y,\ (x + y) + -y = x$:

```
(fun x y ⇒
   (e2l_cor zGroup {1↦x,2↦y}
   (add (add (var 1) (var 2)) (opp (var 2)))))
```

Proof of $\forall x \forall y,\ (x \oplus y) \oplus \neg y = x$:

```
(fun x y ⇒
   (e2l_cor oGroup {1↦x,2↦y}
   (add (add (var 1) (var 2)) (opp (var 2)))))
```

Reflection

Summary:

Reflection

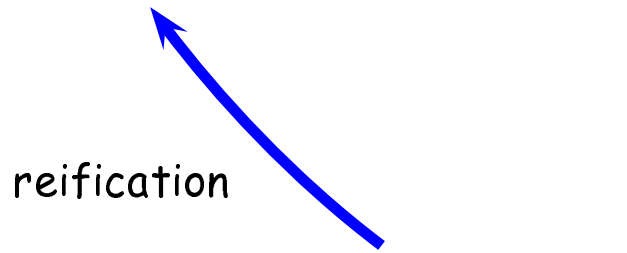
Summary:

$$(x + y) + -y$$

Reflection

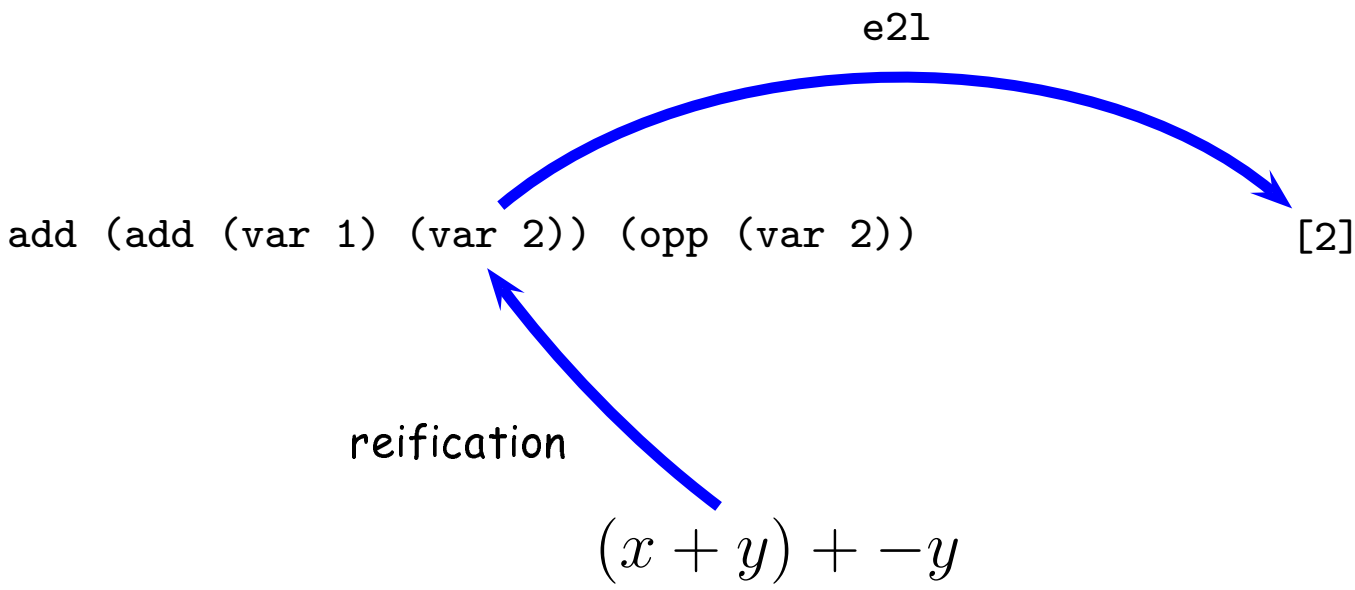
Summary:

```
add (add (var 1) (var 2)) (opp (var 2))
```



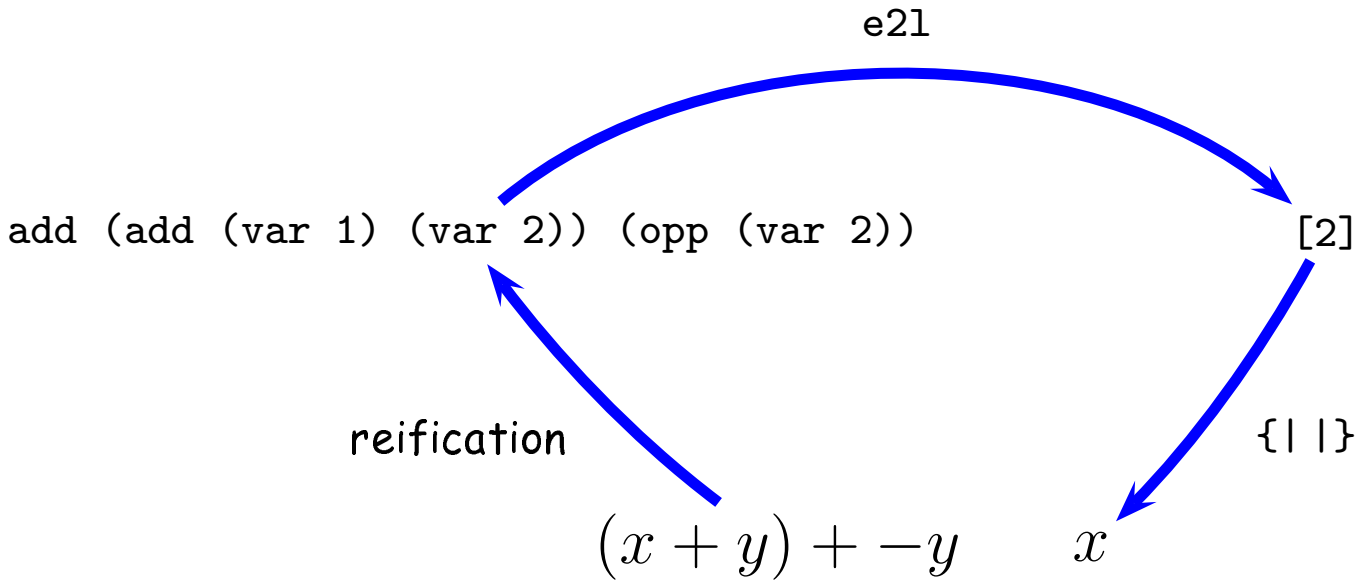
Reflection

Summary:



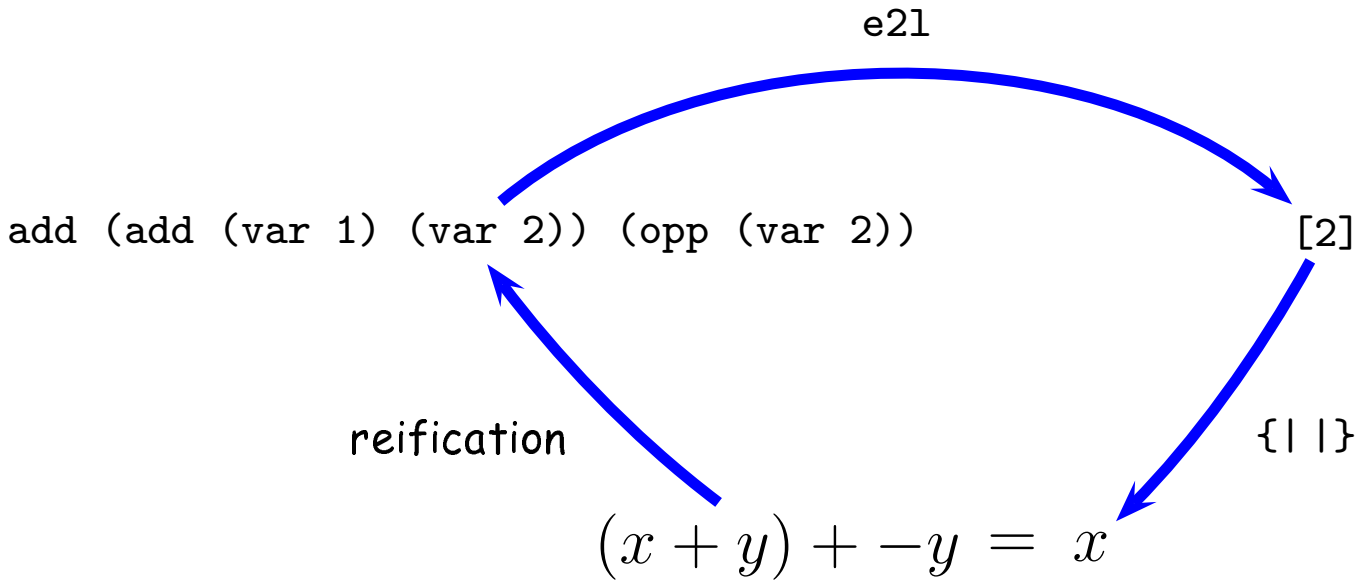
Reflection

Summary:



Reflection

Summary:



Reflection

Degen's identity: ('97) 5m, (v8.2) 0,1 s

Reflection

Degen's identity: ('97) 5m, (v8.2) 0,1 s

Benjamin Grégoire, Assia Mahboubi ('05)
"Proving equalities in a commutative ring done right in Coq"

Reflection

Degen's identity: ('97) 5m, (v8.2) 0,1 s

Benjamin Grégoire, Assia Mahboubi ('05)
"Proving equalities in a commutative ring done right in Coq"

Benjamin Grégoire, Xavier Leroy ('02)
"A compiled implementation of strong reduction"

Extension

Extension

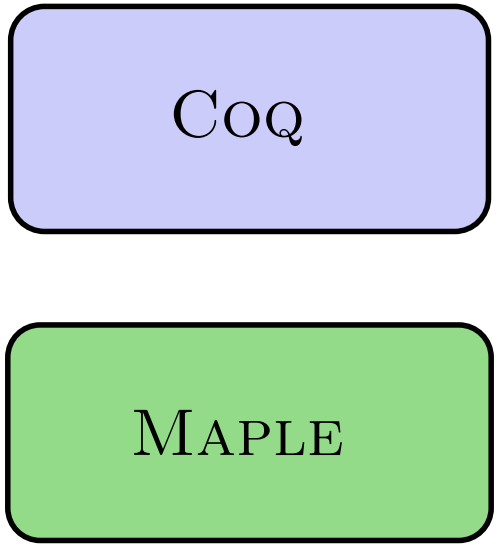
David Delahaye, Micaela Mayero ('05)

" Dealing with algebraic expressions over a field in Coq using Maple"

Extension

David Delahaye, Micaela Mayero ('05)

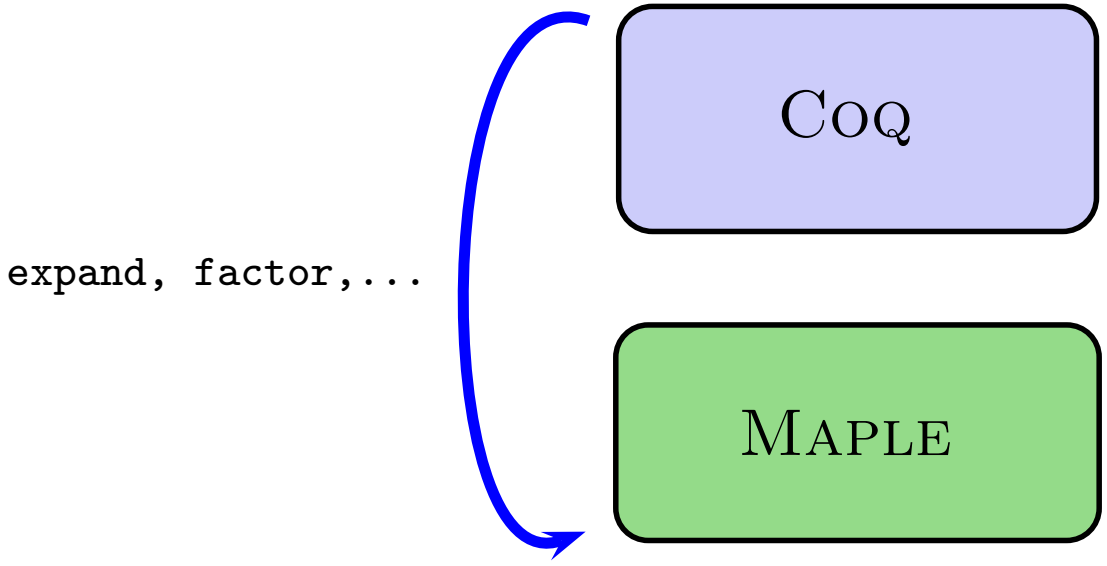
" Dealing with algebraic expressions over a field in Coq using Maple"



Extension

David Delahaye, Micaela Mayero ('05)

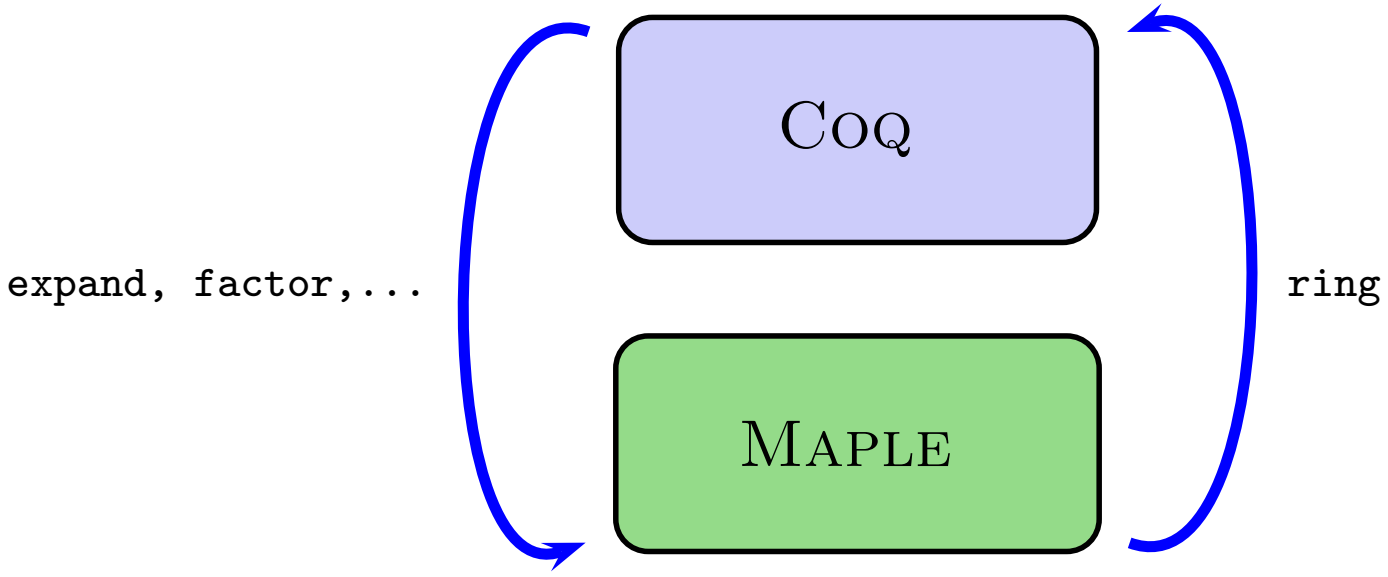
" Dealing with algebraic expressions over a field in Coq using Maple"



Extension

David Delahaye, Micaela Mayero ('05)

" Dealing with algebraic expressions over a field in Coq using Maple"



Extension

Laurent Théry ('05)

"Simplifying Polynomial Expressions in a Proof Assistant"

Extension

Laurent Théry ('05)

"Simplifying Polynomial Expressions in a Proof Assistant"

$$y(3x + y + 2) < x(3y + 1) + z(x + y)$$

Extension

Laurent Théry ('05)
"Simplifying Polynomial Expressions in a Proof Assistant"

$$y(3x + y + 2) < x(3y + 1) + z(x + y)$$

Normalisation:

$$0 < x - yy - 2y + zx + zy$$

Extension

Laurent Théry ('05)
"Simplifying Polynomial Expressions in a Proof Assistant"

$$y(3x + y + 2) < x(3y + 1) + z(x + y)$$

Normalisation:

$$0 < x - yy - 2y + zx + zy$$

Extension

Laurent Théry ('05)
"Simplifying Polynomial Expressions in a Proof Assistant"

$$y(3x + y + 2) < x(3y + 1) + z(x + y)$$

Normalisation:

$$0 < x - yy - 2y + zx + zy$$

Interactive:

$$y(y + 2) < x + z(x + y)$$

Extension

Frédéric Besson ('07)

"Fast Reflexive Arithmetics Tactics the linear case and beyond"

Extension

Frédéric Besson ('07)

"Fast Reflexive Arithmetics Tactics the linear case and beyond"

$$E = c \mid cx \mid E_1 + E_2$$

$$E_1 \leq E_2 \quad E_1 < E_2 \quad E_1 = E_2$$

Extension

Frédéric Besson ('07)

"Fast Reflexive Arithmetics Tactics the linear case and beyond"

$$E = c \mid cx \mid E_1 + E_2$$

$$E_1 \leq E_2 \quad E_1 < E_2 \quad E_1 = E_2$$

$$\begin{cases} -2x - y \geq -1 \\ x + y \geq 2 \\ -y \geq -1 \end{cases}$$

Extension

Farka's Lemma:

Exactly one of the following statements holds:

$$\exists x, A \,.\, x \geq b$$

$$\exists y, y \geq 0 \wedge A^y \,.\, y = 0 \wedge b^t \,.\, y > 0$$

Extension

Farka's Lemma:

Exactly one of the following statements holds:

$$\exists x, A \,.\, x \geq b$$

$$\exists y, y \geq 0 \wedge A^y \,.\, y = 0 \wedge b^t \,.\, y > 0$$

$$\begin{cases} -2x - y \geq -1 \\ x + y \geq 2 \\ -y \geq -1 \end{cases}$$

Extension

Farka's Lemma:

Exactly one of the following statements holds:

$$\exists x, A \,.\, x \geq b$$

$$\exists y, y \geq 0 \wedge A^y \,.\, y = 0 \wedge b^t \,.\, y > 0$$

$$\left\{ \begin{array}{rcrcrcl} -2x & - & y & \geq & -1 & \times & 1 \\ x & + & y & \geq & 2 & \times & 2 \\ & - & y & \geq & -1 & \times & 1 \\ \hline & & 0 & \geq & 2 & & \end{array} \right.$$

Extension

Farka's Lemma:

Exactly one of the following statements holds:

$$\exists x, A \,.\, x \geq b$$

$$\exists y, y \geq 0 \wedge A^y \,.\, y = 0 \wedge b^t \,.\, y > 0$$

$$\left\{ \begin{array}{rcrcrcl} -2x & - & y & \geq & -1 & \times & 1 \\ x & + & y & \geq & 2 & \times & 2 \\ & - & y & \geq & -1 & \times & 1 \\ \hline & & 0 & \geq & 2 & & \end{array} \right.$$

Extension

John Harrison ('07)

"Verifying nonlinear real formulas via sums of square"

Extension

John Harrison ('07)

"Verifying nonlinear real formulas via sums of square"

$$\forall x_1, \ldots, x_n,$$
$$P_1(x_1, \ldots, x_n) \geq 0 \quad \wedge \quad \ldots \quad \wedge \quad P_k(x_1, \ldots, x_n) \geq 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n) \geq 0$$

Extension

Basic Idea:

$$P(x_1, \ldots, x_n)^2 \geq 0$$

$$P(x_1, \ldots, x_n) \geq 0 \quad \wedge \quad P(x_1, \ldots, x_n) \geq 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n) + Q(x_1, \ldots, x_n) \geq 0$$

$$P(x_1, \ldots, x_n) \geq 0 \quad \wedge \quad Q(x_1, \ldots, x_n) \geq 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n)\, Q(x_1, \ldots, x_n) \geq 0$$

Extension

Basic Idea:

$$P(x_1, \ldots, x_n)^2 \geq 0$$

$$P(x_1, \ldots, x_n) \geq 0 \quad \wedge \quad P(x_1, \ldots, x_n) \geq 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n) + Q(x_1, \ldots, x_n) \geq 0$$

$$P(x_1, \ldots, x_n) \geq 0 \quad \wedge \quad Q(x_1, \ldots, x_n) \geq 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n)\, Q(x_1, \ldots, x_n) \geq 0$$

Example

$$\forall a\, b\, c\, x, \quad ax^2 + bx + c = 0 \quad \Rightarrow \quad b^2 - 4ac \geq 0$$

Extension

Basic Idea:

$$P(x_1, \ldots, x_n)^2 \geq 0$$

$$P(x_1, \ldots, x_n) \geq 0 \quad \wedge \quad P(x_1, \ldots, x_n) \geq 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n) + Q(x_1, \ldots, x_n) \geq 0$$

$$P(x_1, \ldots, x_n) \geq 0 \quad \wedge \quad Q(x_1, \ldots, x_n) \geq 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n)\, Q(x_1, \ldots, x_n) \geq 0$$

Example

$$\forall a\, b\, c\, x, \quad ax^2 + bx + c = 0 \quad \Rightarrow \quad b^2 - 4ac \geq 0$$

$$b^2 - 4ac =$$

Extension

Basic Idea:

$$P(x_1, \ldots, x_n)^2 \geq 0$$

$$P(x_1, \ldots, x_n) \geq 0 \quad \wedge \quad P(x_1, \ldots, x_n) \geq 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n) + Q(x_1, \ldots, x_n) \geq 0$$

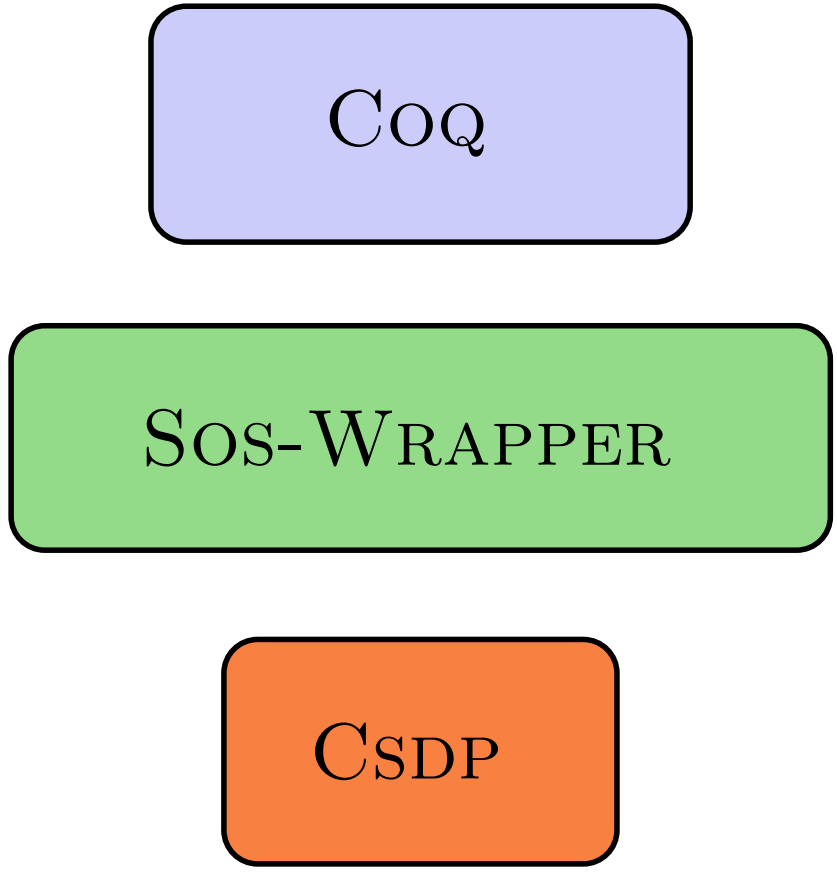
$$P(x_1, \ldots, x_n) \geq 0 \quad \wedge \quad Q(x_1, \ldots, x_n) \geq 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n)\, Q(x_1, \ldots, x_n) \geq 0$$

Example

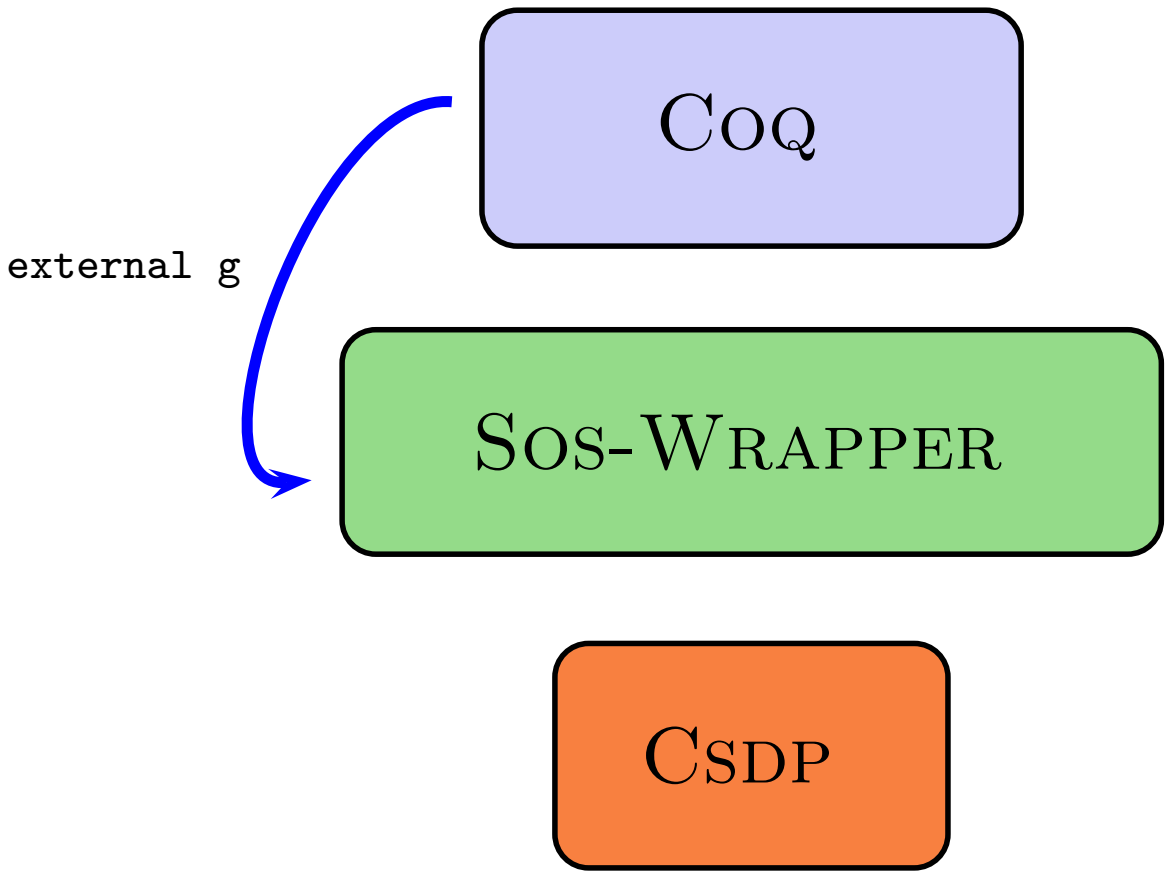
$$\forall a\, b\, c\, x, \quad ax^2 + bx + c = 0 \quad \Rightarrow \quad b^2 - 4ac \geq 0$$

$$b^2 - 4ac = (2ax + b)^2 - 4a(ax^2 + bx + c)$$

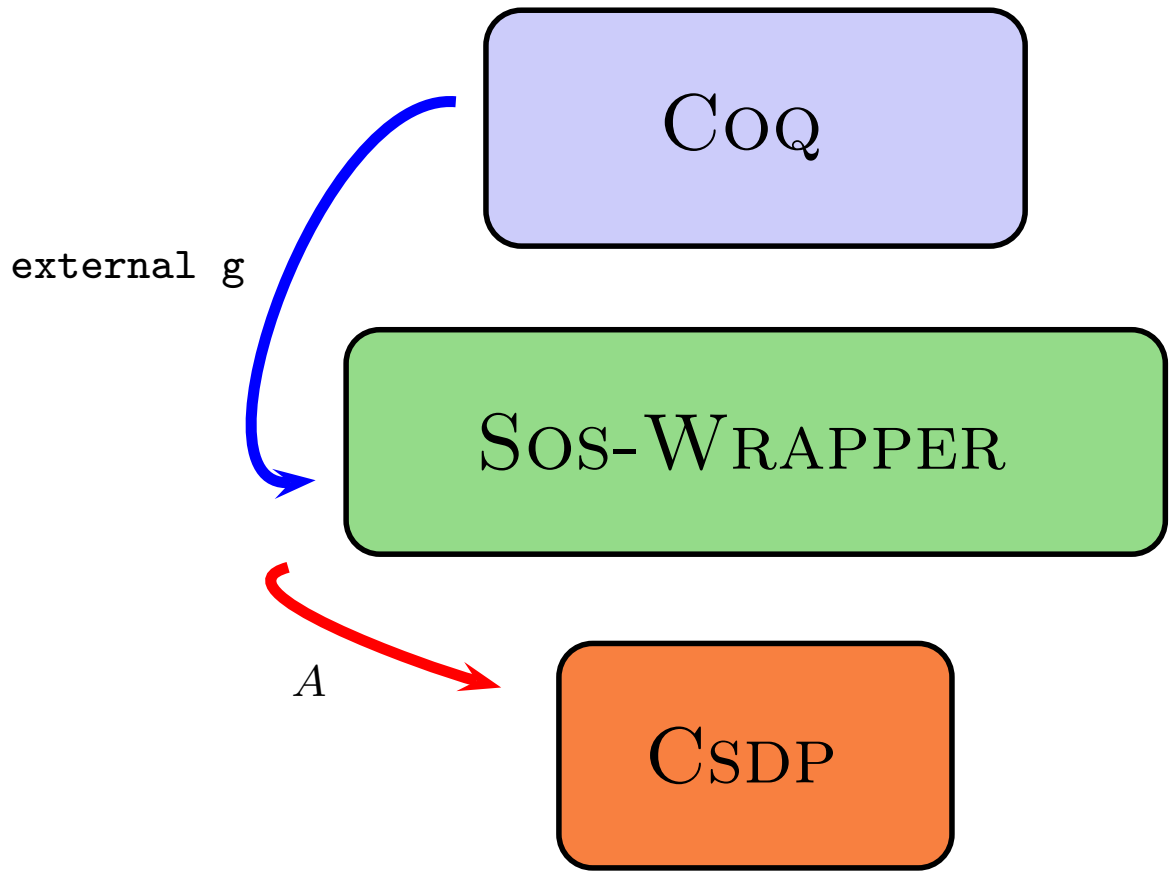
Extension



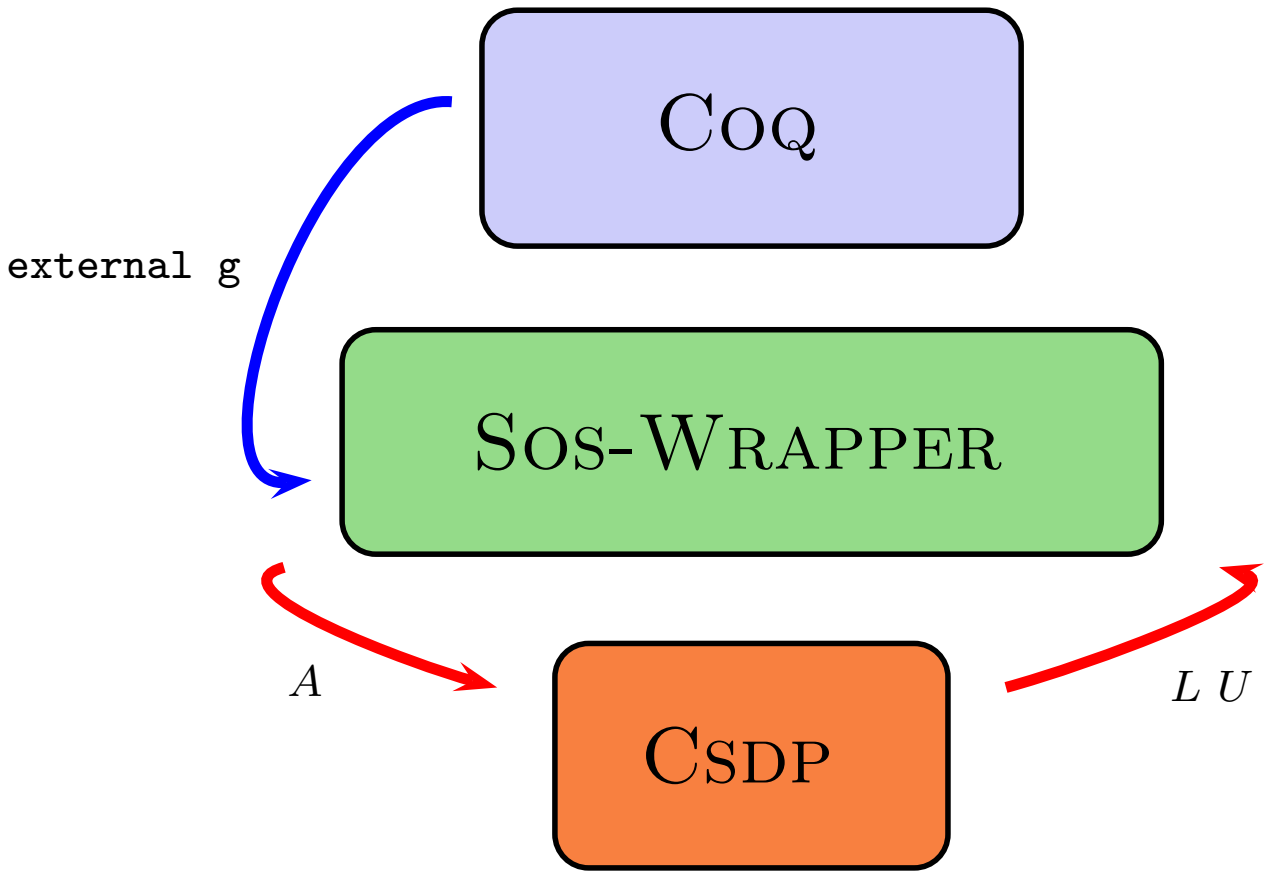
Extension



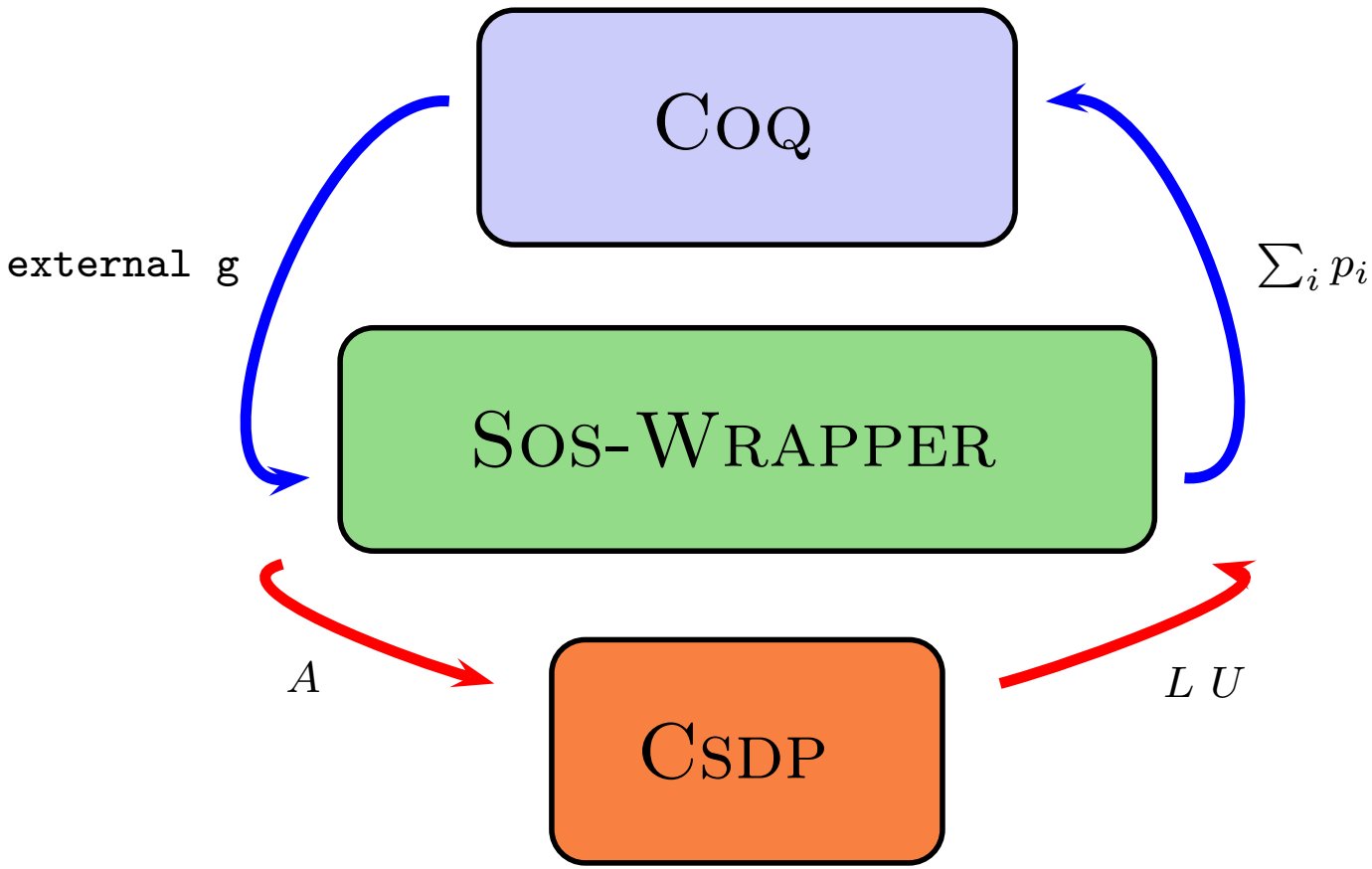
Extension



Extension



Extension



Extension

Robert S. Boyer, J Strother Moore ('81)

"The Mechanical Verification of a Fortran Square Root Program"

Extension

Robert S. Boyer, J Strother Moore ('81)

"The Mechanical Verification of a Fortran Square Root Program"

```
      INTEGER FUNCTION ISQRT(I)
      INTEGER I
      IF ((I .LT. 0)) STOP
      IF ((I .GT. 1)) GOTO 100
      ISQRT = I
      RETURN
100   ISQRT = (I / 2)
200   CONTINUE
      IF (((I / ISQRT) .GE. ISQRT)) RETURN
      ISQRT = ((ISQRT + (I / ISQRT)) / 2)
      GOTO 200
      END
```

Extension

Main theorem:

$$\forall i\, j,\ 0 \leq i \wedge 0 < j \Rightarrow i < ((j + i/j)/2 + 1)^2$$

Extension

Main theorem:

$$\forall i\, j,\ 0 \leq i \wedge 0 < j \Rightarrow i < ((j + i/j)/2 + 1)^2$$

Reduction: $k = i/j$ and $l = (j + k)/2$

$$\begin{array}{l} \forall j\, k\, l, \\ \quad 0 \leq j \wedge 0 \leq k \wedge 0 \leq l \wedge 2l \leq j{+}k \wedge j + k < 2(l + 1) \\ \qquad \Rightarrow\ jk + j \leq (l + 1)^2 \end{array}$$

Extension

Main theorem:

$$\forall i\, j,\ 0 \leq i \wedge 0 < j \Rightarrow i < ((j + i/j)/2 + 1)^2$$

Reduction: $k = i/j$ and $l = (j + k)/2$

$$\begin{array}{l}\forall j\, k\, l,\\ \quad 0 \leq j \wedge 0 \leq k \wedge 0 \leq l \wedge 2l \leq j{+}k \wedge j + k < 2(l + 1)\\ \qquad \Rightarrow\ jk + j \leq (l + 1)^2\end{array}$$

Sum of Squares:

$$\begin{array}{l}4((l + 1)^2 - (jk + j)) =\\ (k - j + 1)^2 + (2(l + 1) - (j + k + 1))(3 + j + k + 2l))\end{array}$$

Extension

Benjamin Grégoire, Loïc Pottier, Laurent Théry ('09)
"Proofs Certificates for Algebra and their Application to Automatic Geometry Theorem Proving"

Extension

Benjamin Grégoire, Loïc Pottier, Laurent Théry ('09)
"Proofs Certificates for Algebra and their Application to Automatic Geometry Theorem Proving"

$$\forall x_1, \ldots, x_n,$$
$$P_1(x_1, \ldots, x_n) = 0 \quad \wedge \quad \ldots \quad \wedge \quad P_k(x_1, \ldots, x_n) = 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n) = 0$$

Extension

Benjamin Grégoire, Loïc Pottier, Laurent Théry ('09)
"Proofs Certificates for Algebra and their Application to Automatic Geometry Theorem Proving"

$$\forall x_1, \ldots, x_n,$$
$$P_1(x_1, \ldots, x_n) = 0 \quad \wedge \quad \ldots \quad \wedge \quad P_k(x_1, \ldots, x_n) = 0 \quad \Rightarrow$$
$$P(x_1, \ldots, x_n) = 0$$

Hilbert's Nullstellensatz

$$cP(x_1, \ldots, x_n)^r = \sum_i Q_i(x_1, \ldots, x_n) P_i(x_1, \ldots, x_n)$$

Extension

Basic Idea

Extension

Basic Idea

$$P(x_1, \ldots, x_n)^r \in \{P_1(x_1, \ldots, x_n), \ldots, P_k(x_1, \ldots, x_n)\}$$

Extension

Basic Idea

$$P(x_1, \ldots, x_n)^r \in \{P_1(x_1, \ldots, x_n), \ldots, P_k(x_1, \ldots, x_n)\}$$

Example

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$

Division

$$x^2y^2 - y^4 \qquad x^2 + 1$$

Extension

Basic Idea

$$P(x_1, \ldots, x_n)^r \in \{P_1(x_1, \ldots, x_n), \ldots, P_k(x_1, \ldots, x_n)\}$$

Example

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$

Division

$x^2y^2 - y^4$ $\qquad$ $x^2 + 1$

Extension

Basic Idea

$$P(x_1, \ldots, x_n)^r \in \{P_1(x_1, \ldots, x_n), \ldots, P_k(x_1, \ldots, x_n)\}$$

Example

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$

Division

$$x^2y^2 - y^4 = y^2(x^2 + 1) - y^4 - y^2$$

Extension

Basic Idea

$$P(x_1, \ldots, x_n)^r \in \{P_1(x_1, \ldots, x_n), \ldots, P_k(x_1, \ldots, x_n)\}$$

Example

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$

Division

$$x^2y^2 - y^4 = y^2(x^2 + 1) - y^4 - y^2$$

$$x^2y^2 - y^4 = (xy + 1)(xy - 1) - y^4 + 1$$

Extension

Basic Idea

$$P(x_1, \ldots, x_n)^r \in \{P_1(x_1, \ldots, x_n), \ldots, P_k(x_1, \ldots, x_n)\}$$

Example

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$

Division

$$x^2y^2 - y^4 = y^2(x^2 + 1) - y^4 - y^2$$

$$x^2y^2 - y^4 = (xy + 1)(xy - 1) - y^4 + 1$$

Spolynomial

$$x^2y = y(x^2 + 1) = x(xy - 1)$$

$$spoly(x^2 + 1, xy + 1) = y(x^2 + 1) - x(xy - 1) = x + y$$

Extension

Gröbner basis

$$\{x^2 + 1, xy - 1\}$$

$$spoly(x^2 + 1, xy + 1) = x + y$$

Extension

Gröbner basis

$$\{x^2 + 1, xy - 1\}$$

$$spoly(x^2 + 1, xy + 1) = x + y$$

$$\{x^2 + 1, xy - 1, x + y\}$$

Extension

Gröbner basis

$$\{x^2 + 1, xy - 1\}$$

$$spoly(x^2 + 1, xy + 1) = x + y$$

$$\{x^2 + 1, xy - 1, x + y\}$$

$$spoly(x^2 + 1, x + y) = x^2 + 1 - x(x + y) = -(xy - 1)$$

Extension

Gröbner basis

$$\{x^2+1, xy-1\}$$

$$spoly(x^2+1, xy+1) = x+y$$

$$\{x^2+1, xy-1, x+y\}$$

$$spoly(x^2+1, x+y) = x^2+1-x(x+y) = -(xy-1)$$
$$spoly(xy-1, x+y) = xy-1-y(x+y) = -y^2-1$$

Extension

Gröbner basis

$$\{x^2+1, xy-1\}$$

$$spoly(x^2+1, xy+1) = x+y$$

$$\{x^2+1, xy-1, x+y\}$$

$$spoly(x^2+1, x+y) = x^2+1-x(x+y) = -(xy-1)$$
$$spoly(xy-1, x+y) = xy-1-y(x+y) = -y^2-1$$

$$\{x^2+1, xy-1, x+y, -y^2-1\}$$

Extension

Gröbner basis

$$\{x^2+1, xy-1\}$$

$$spoly(x^2+1, xy+1) = x+y$$

$$\{x^2+1, xy-1, x+y\}$$

$$spoly(x^2+1, x+y) = x^2+1-x(x+y) = -(xy-1)$$
$$spoly(xy-1, x+y) = xy-1-y(x+y) = -y^2-1$$

$$\{x^2+1, xy-1, x+y, -y^2-1\}$$

$$spoly(xy-1, -y^2-1) = y(xy-1)-(-x)(-y^2-1) = -(x+y)$$

Extension

Certificate

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$

Extension

Certificate

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$
$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1, x + y, -y^2 - 1\}$$

Extension

Certificate

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$
$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1, x + y, -y^2 - 1\}$$

$$x^2y^2 - y^4 = y^2(x^2 + 1) - y^4 - y^2$$

Extension

Certificate

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$
$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1, x + y, -y^2 - 1\}$$

$$x^2y^2 - y^4 = y^2(x^2 + 1) - y^4 - y^2$$
$$-y^4 - y^2 = y^2(-y^2 - 1) + 0$$

Extension

Certificate

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$
$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1, x + y, -y^2 - 1\}$$

$$x^2y^2 - y^4 = y^2(x^2 + 1) - y^4 - y^2$$
$$-y^4 - y^2 = y^2(-y^2 - 1) + 0$$

$$x^2y^2 - y^4 = y^2(x^2 + 1) + y^2(-y^2 - 1)$$

Extension

Certificate

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$
$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1, x + y, -y^2 - 1\}$$

$$x^2y^2 - y^4 = y^2(x^2 + 1) - y^4 - y^2$$
$$-y^4 - y^2 = y^2(-y^2 - 1) + 0$$

$$x^2y^2 - y^4 = y^2(x^2 + 1) + y^2(-y^2 - 1)$$
$$x^2y^2 - y^4 = y^2(x^2 + 1) + y^2(xy - 1 - y(x + y))$$

Extension

Certificate

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$
$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1, x + y, -y^2 - 1\}$$

$$x^2y^2 - y^4 = y^2(x^2 + 1) - y^4 - y^2$$
$$-y^4 - y^2 = y^2(-y^2 - 1) + 0$$

$$x^2y^2 - y^4 = y^2(x^2 + 1) + y^2(-y^2 - 1)$$
$$x^2y^2 - y^4 = y^2(x^2 + 1) + y^2(xy - 1 - y(x + y))$$
$$x^2y^2 - y^4 = y^2(x^2+1) + y^2(xy - 1 - y(y(x^2+1) - x(xy - 1)))$$

Extension

Certificate

$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1\}$$
$$x^2y^2 - y^4 \in \{x^2 + 1, xy - 1, x + y, -y^2 - 1\}$$

$$x^2y^2 - y^4 = y^2(x^2 + 1) - y^4 - y^2$$
$$-y^4 - y^2 = y^2(-y^2 - 1) + 0$$

$$x^2y^2 - y^4 = y^2(x^2 + 1) + y^2(-y^2 - 1)$$
$$x^2y^2 - y^4 = y^2(x^2 + 1) + y^2(xy - 1 - y(x + y))$$
$$x^2y^2 - y^4 = y^2(x^2 + 1) + y^2(xy - 1 - y(y(x^2 + 1) - x(xy - 1)))$$
$$x^2y^2 - y^4 = (-y^4 + y^2)(x^2 + 1) + (xy^3 + y^2)(xy - 1)$$

Extension

Straight-Line Program

Extension

Straight-Line Program

$$X^{f_{n+1}} - 1 = 0 \wedge X^{f_n} - 1 = 0 \Rightarrow X - 1 = 0$$

where f_n: $f_0 = 0, \quad f_1 = 1, \quad f_{n+2} = f_{n+1} + f_n$

Extension

Straight-Line Program

$$X^{f_{n+1}} - 1 = 0 \wedge X^{f_n} - 1 = 0 \Rightarrow X - 1 = 0$$

where f_n: $f_0 = 0, \quad f_1 = 1, \quad f_{n+2} = f_{n+1} + f_n$

$$X - 1 = P_{n-2}(X^{f_{n+1}} - 1) + P_{n-1}(X^{f_n} - 1)$$

where P_n: $P_0 = 0, P_1 = 1, P_n = -X^{f_n} P_{n-1} + P_{n-2}$

Extension

Straight-Line Program

$$X^{f_{n+1}} - 1 = 0 \wedge X^{f_n} - 1 = 0 \Rightarrow X - 1 = 0$$

where f_n: $f_0 = 0, \quad f_1 = 1, \quad f_{n+2} = f_{n+1} + f_n$

$$X - 1 = P_{n-2}(X^{f_{n+1}} - 1) + P_{n-1}(X^{f_n} - 1)$$

where P_n: $P_0 = 0, P_1 = 1, P_n = -X^{f_n} P_{n-1} + P_{n-2}$

$$\begin{aligned} C = [&[1, -X^{f_{n-1}}] \\ &[0, 1, -X^{f_{n-2}}], \\ &\ldots \\ &[0, \ldots, 0, 1, -X^{f_2}]] \end{aligned}$$

$$CR = [0, \ldots, 0, 1]$$

Extension

Straight-Line Program

$$X^{f_{n+1}} - 1 = 0 \wedge X^{f_n} - 1 = 0 \Rightarrow X - 1 = 0$$

where f_n: $f_0 = 0, \quad f_1 = 1, \quad f_{n+2} = f_{n+1} + f_n$

$$X - 1 = P_{n-2}(X^{f_{n+1}} - 1) + P_{n-1}(X^{f_n} - 1)$$

where P_n: $P_0 = 0, P_1 = 1, P_n = -X^{f_n} P_{n-1} + P_{n-2}$

$$P_1 = X^{f_{n+1}} - 1 \quad P_2 = X^{f_n} - 1$$

$$\begin{aligned} C = [&[1, -X^{f_{n-1}}] \\ &[0, 1, -X^{f_{n-2}}], \\ &\ldots \\ &[0, \ldots, 0, 1, -X^{f_2}]] \end{aligned}$$

$$CR = [0, \ldots, 0, 1]$$

Extension

Straight-Line Program

$$X^{f_{n+1}} - 1 = 0 \wedge X^{f_n} - 1 = 0 \Rightarrow X - 1 = 0$$

where f_n: $f_0 = 0, \quad f_1 = 1, \quad f_{n+2} = f_{n+1} + f_n$

$$X - 1 = P_{n-2}(X^{f_{n+1}} - 1) + P_{n-1}(X^{f_n} - 1)$$

where P_n: $P_0 = 0, P_1 = 1, P_n = -X^{f_n} P_{n-1} + P_{n-2}$

$$P_1 = X^{f_{n+1}} - 1 \quad P_2 = X^{f_n} - 1$$

$$P_3 = P_1 - X^{f_{n-1}} P_2$$

$$C = [[1, -X^{f_{n-1}}]$$
$$[0, 1, -X^{f_{n-2}}],$$
$$\ldots$$
$$[0, \ldots, 0, 1, -X^{f_2}]]$$
$$CR = [0, \ldots, 0, 1]$$

Extension

Straight-Line Program

$$X^{f_{n+1}} - 1 = 0 \wedge X^{f_n} - 1 = 0 \Rightarrow X - 1 = 0$$

where f_n: $f_0 = 0, \quad f_1 = 1, \quad f_{n+2} = f_{n+1} + f_n$

$$X - 1 = P_{n-2}(X^{f_{n+1}} - 1) + P_{n-1}(X^{f_n} - 1)$$

where P_n: $P_0 = 0, P_1 = 1, P_n = -X^{f_n} P_{n-1} + P_{n-2}$

$$P_1 = X^{f_{n+1}} - 1 \quad P_2 = X^{f_n} - 1$$

$$C = [[1, -X^{f_{n-1}}] \qquad P_3 = P_1 - X^{f_{n-1}} P_2$$

$$[0, 1, -X^{f_{n-2}}], \qquad P_4 = P_2 - X^{f_{n-2}} P_3$$

$$\ldots$$

$$[0, \ldots, 0, 1, -X^{f_2}]]$$

$$CR = [0, \ldots, 0, 1]$$

Extension

Straight-Line Program

$$X^{f_{n+1}} - 1 = 0 \wedge X^{f_n} - 1 = 0 \Rightarrow X - 1 = 0$$

where f_n: $f_0 = 0, \quad f_1 = 1, \quad f_{n+2} = f_{n+1} + f_n$

$$X - 1 = P_{n-2}(X^{f_{n+1}} - 1) + P_{n-1}(X^{f_n} - 1)$$

where P_n: $P_0 = 0, P_1 = 1, P_n = -X^{f_n} P_{n-1} + P_{n-2}$

$$C = [[1, -X^{f_{n-1}}] \qquad P_1 = X^{f_{n+1}} - 1 \quad P_2 = X^{f_n} - 1$$
$$[0, 1, -X^{f_{n-2}}], \qquad P_3 = P_1 - X^{f_{n-1}} P_2$$
$$\ldots \qquad P_4 = P_2 - X^{f_{n-2}} P_3$$
$$[0, \ldots, 0, 1, -X^{f_2}]] \qquad \ldots$$
$$CR = [0, \ldots, 0, 1] \qquad P_n = P_{n-2} - X^{f_2} P_{n-1}$$

Extension

Straight-Line Program

$$X^{f_{n+1}} - 1 = 0 \wedge X^{f_n} - 1 = 0 \Rightarrow X - 1 = 0$$

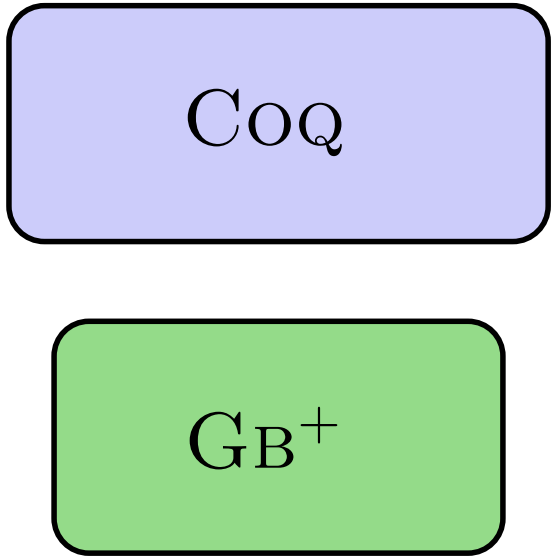
where f_n: $f_0 = 0, \quad f_1 = 1, \quad f_{n+2} = f_{n+1} + f_n$

$$X - 1 = P_{n-2}(X^{f_{n+1}} - 1) + P_{n-1}(X^{f_n} - 1)$$

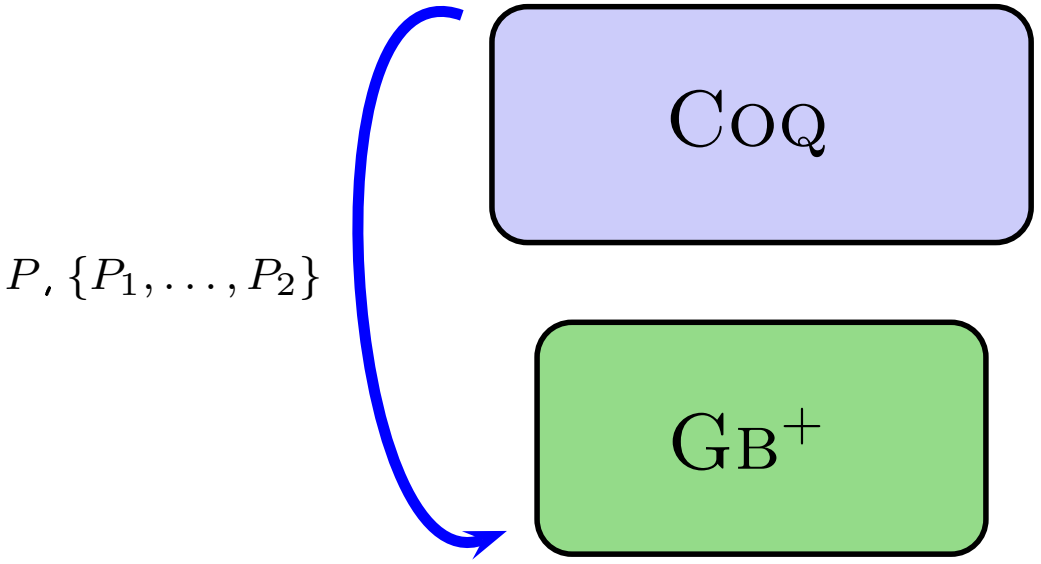
where P_n: $P_0 = 0, P_1 = 1, P_n = -X^{f_n} P_{n-1} + P_{n-2}$

$$\begin{array}{ll} & P_1 = X^{f_{n+1}} - 1 \quad P_2 = X^{f_n} - 1 \\ C = [[1, -X^{f_{n-1}}] & P_3 = P_1 - X^{f_{n-1}} P_2 \\ \quad [0, 1, -X^{f_{n-2}}], & P_4 = P_2 - X^{f_{n-2}} P_3 \\ \quad \ldots & \quad \ldots \\ \quad [0, \ldots, 0, 1, -X^{f_2}]] & P_n = P_{n-2} - X^{f_2} P_{n-1} \\ CR = [0, \ldots, 0, 1] & X - 1 = P_n \end{array}$$

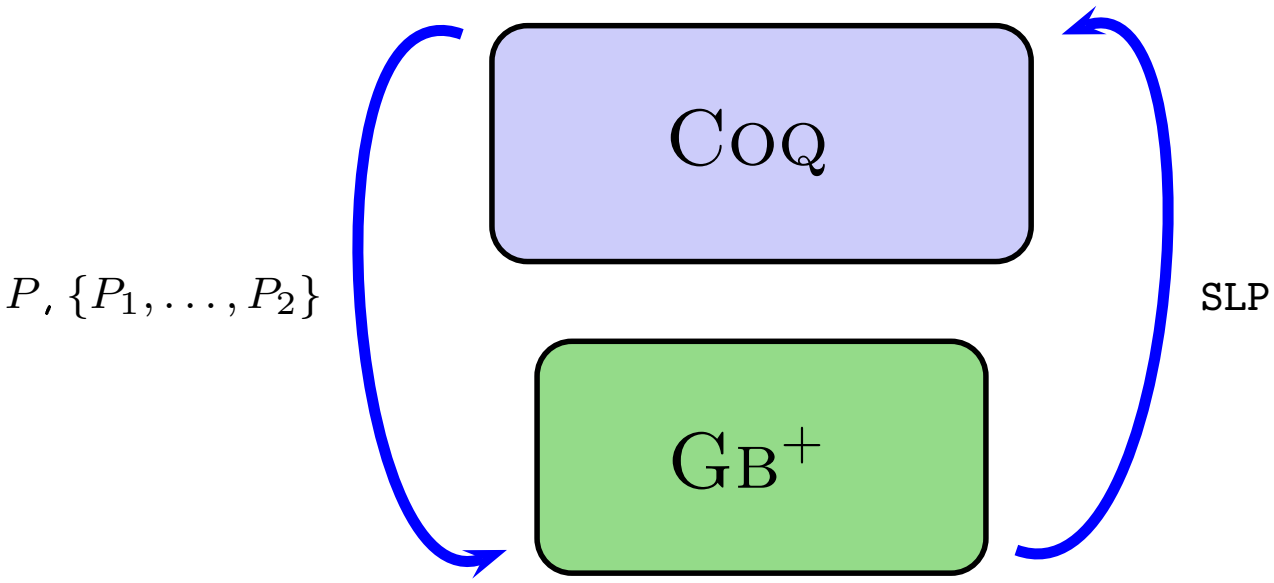
Extension



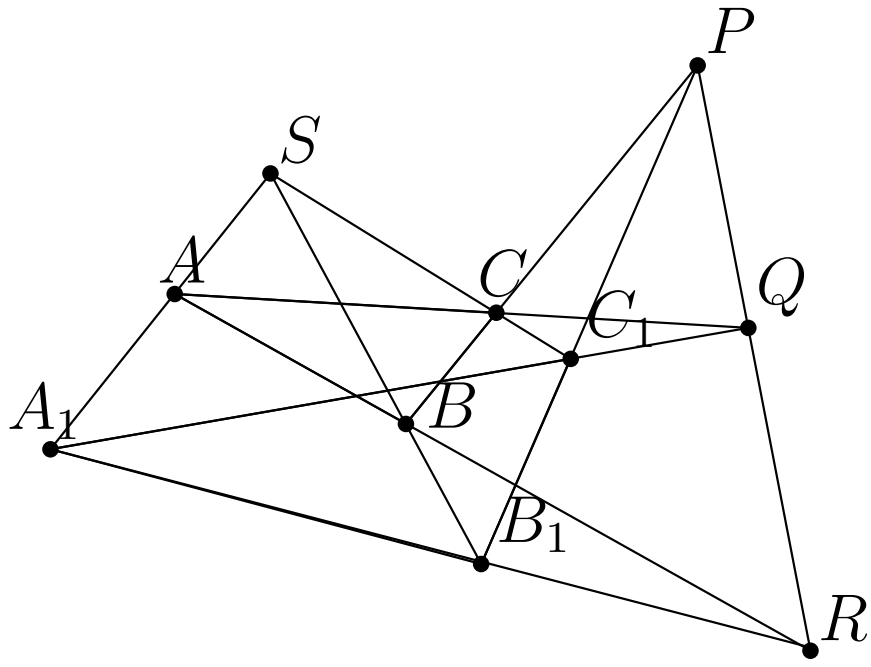
Extension



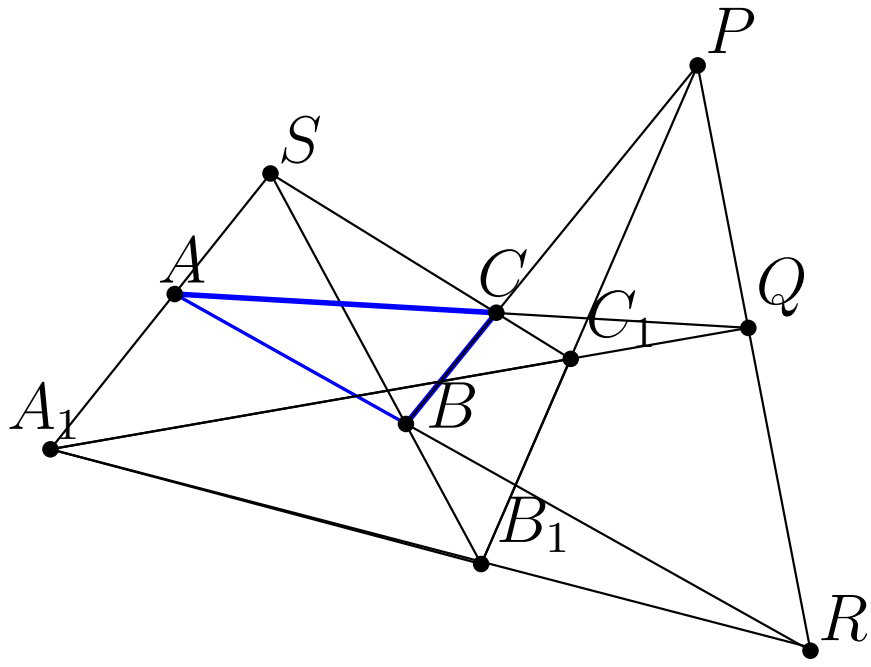
Extension



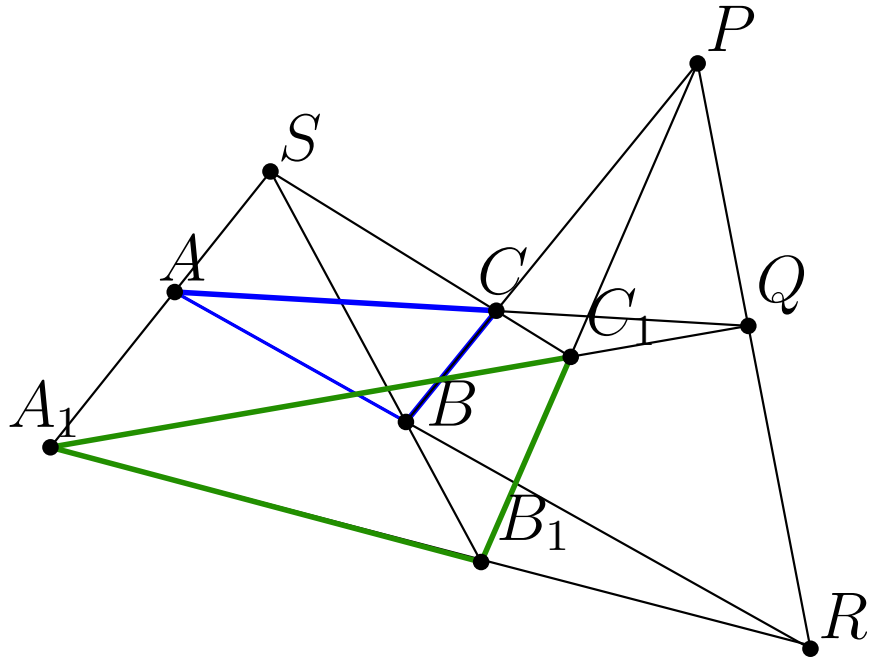
Extension



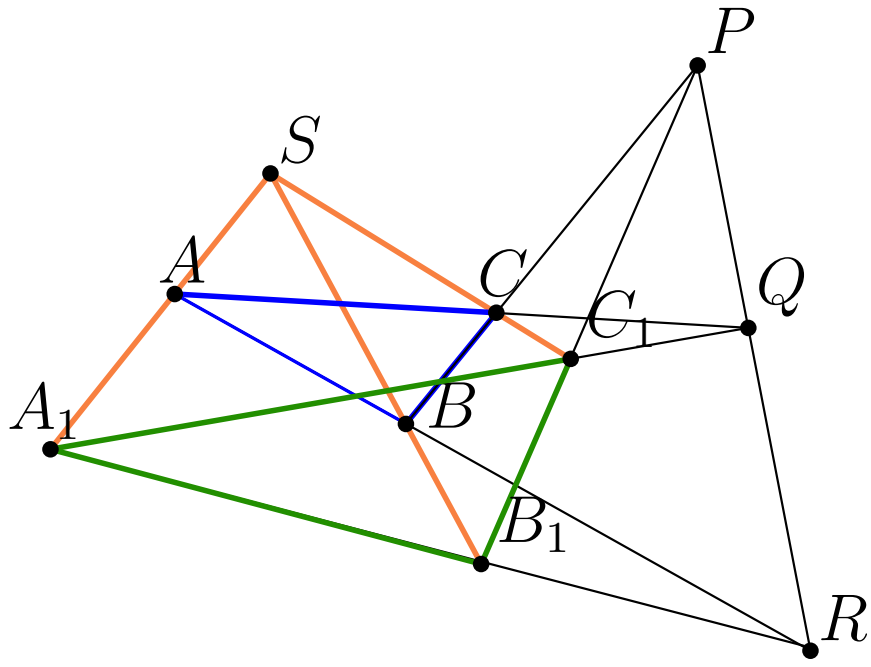
Extension



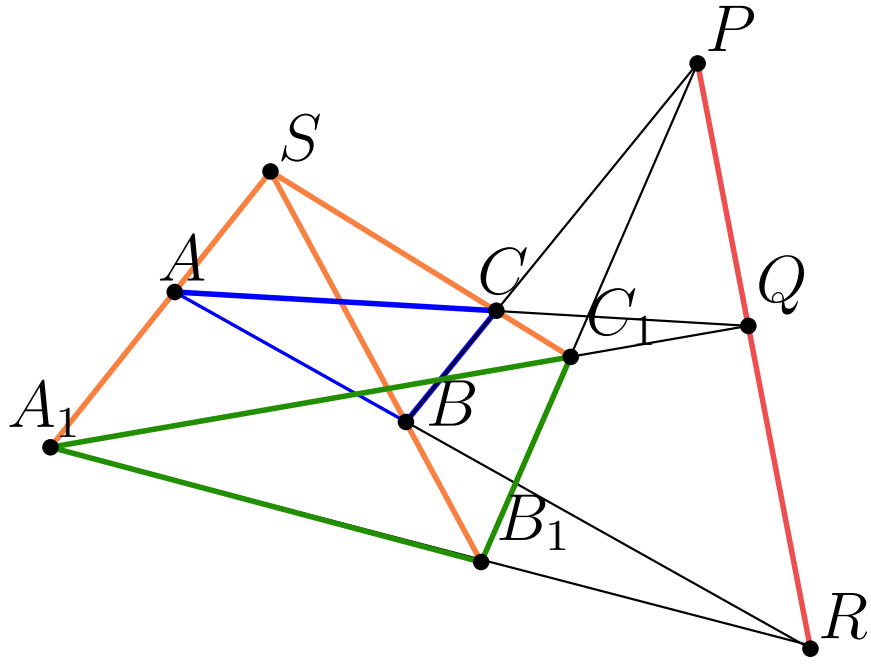
Extension



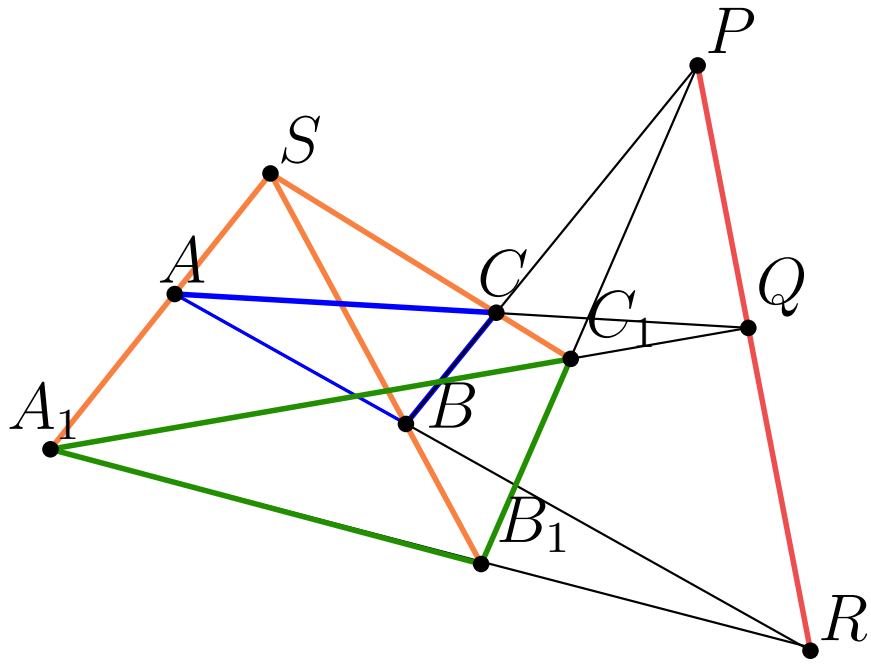
Extension



Extension



Extension




```
Record point := {x:  ℝ, y:  ℝ}.
Definition collinear (A B C:point):=
  (A.x - B.x) * (C.y - B.y) - (A.y - B.y) * (C.x - B.x) = 0.
Definition parallel (A B C D:point):=
  (A.x - B.x) * (C.y - D.y) = (A.y - B.y) * (C.x - D.x).
```

Extension

	Time (seconds)		Size (characters)		Size (nodes)
Theorem	Computing	Verifying	Polynomials	Certificate	SLP
Ceva	181	2.5	538644	477414	76669
Desargues	0.3	0.01	6359	4551	4311
Feuerbach	0.8	0.4	52569	16999	5497
Pappus	1.3	0.2	2721	1934	8031
Pascal	397	12	732982	864509	183505
Ptolemy	200	2.4	571931	571931	73257
Simson	0.3	0.2	1541	1238	4919
Thales	0.03	0.1	5422	5169	1323

More Extensions

More Extensions

Michaël Armand, Benjamin Grégoire, Arnaud Spiwack and Laurent Théry ('10)

"Extending Coq with Imperative Features and its Application to SAT Verification"

More Extensions

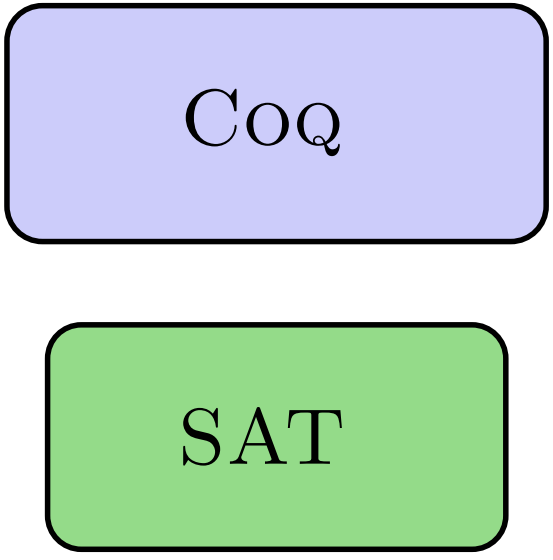
Michaël Armand, Benjamin Grégoire, Arnaud Spiwack and Laurent Théry ('10)

"Extending Coq with Imperative Features and its Application to SAT Verification"

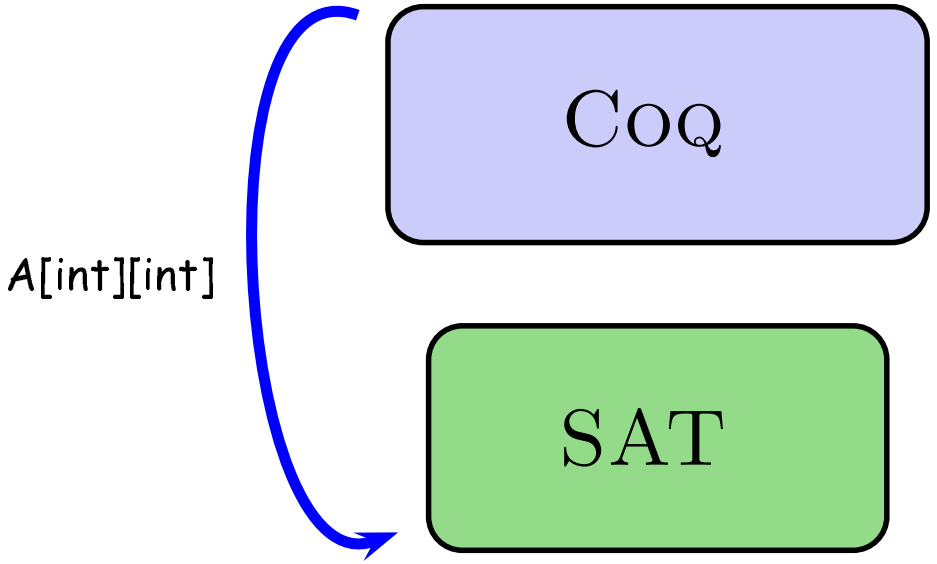
Chantal Keller and Benjamin Werner ('10)

"Importing HOL Light into Coq"

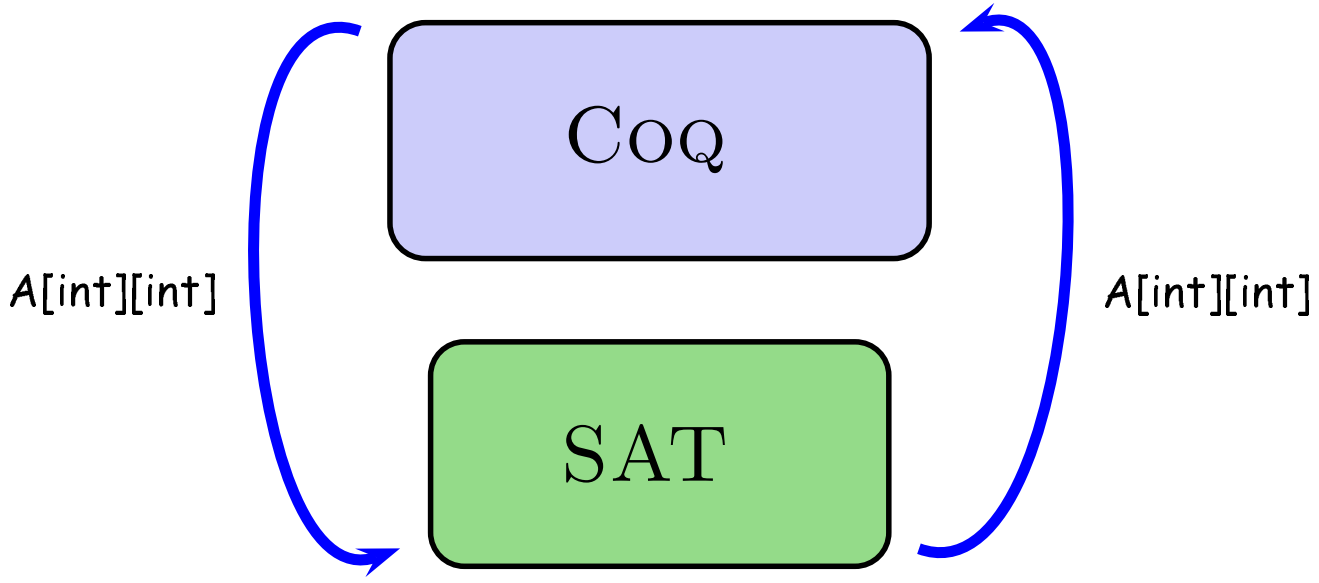
Extension



Extension



Extension



More Extensions

More Extensions

```
p cnf 50 80
 16   23   42 0
-16   23   42 0
 26   41  -42 0
-26   41  -42 0
 32  -41  -42 0
  6   15  -41 0
 -6   15  -32 0
  1  -32   46 0
 -1  -32   46 0
-15  -41  -46 0
-15  -21  -46 0
-23   33   38 0
-23  -33   38 0
  8   22   33 0
  8   22  -33 0
-22   37  -38 0
 13   36  -37 0
 13  -22  -36 0
-13  -22  -37 0
 11  -23   47 0
 -8   11  -47 0
 -8  -11   39 0
-11   27  -39 0
 -8  -11  -39 0
 -7   26   29 0
 -7  -26   29 0
-13   20   36 0
-13   17   20 0
  5  -17   20 0
  5  -19  -45 0
 -5  -10  -45 0
  6   25   47 0
 -6  -10   25 0
 -2  -27   37 0
-27  -36   40 0
 18   39  -40 0
 -2  -19   31 0
  5   18  -30 0
-31  -43  -50 0
 10  -30   43 0
 10  -41   43 0
 19   21   29 0
 37   42   45 0
-20   27   40 0
-21  -36   48 0
 31  -36  -48 0
  3   -9  -18 0
 16  -40  -47 0
  1  -18   21 0
  2   28   32 0
 -1  -24  -50 0
-12   35   49 0
 -6  -36   45 0
  7   12  -43 0
  7   30  -43 0
 -5    9  -17 0
  3   14   50 0
-12   17  -49 0
 24   34   49 0
 14  -20   24 0
 -9   35  -49 0
 -4  -47   50 0
  4   44  -44 0
 28  -28  -38 0
  2    4  -48 0
-20   35  -44 0
 30  -31  -43 0
-14  -29   35 0
-20   35  -35 0
 19  -22  -24 0
 25  -28   48 0
-14  -34   44 0
  9   20   44 0
 -3    9  -29 0
 17   34  -34 0
 12   48   48 0
-12  -25  -43 0
-25  -31   48 0
 14  -16   49 0
 -3   -4  -35
```

More Extensions

```
p cnf 50 80
 16   23   42 0
-16   23   42 0
 26   41  -42 0
-26   41  -42 0
 32  -41  -42 0
  6   15  -41 0
 -6   15  -32 0
  1  -32   46 0
 -1  -32   46 0
-15  -41  -46 0
-15  -21  -46 0
-23   33   38 0
-23  -33   38 0
  8   22   33 0
  8   22  -33 0
-22   37  -38 0
 13   36  -37 0
 13  -22  -36 0
-13  -22  -37 0
 11  -23   47 0
 -8   11  -47 0
 -8  -11   39 0
-11   27  -39 0
 -8  -11  -39 0
 -7   26   29 0
 -7  -26   29 0
-13   20   36 0
-13   17   20 0
  5  -17   20 0
  5  -19  -45 0
 -5  -10  -45 0
  6   25   47 0
 -6  -10   25 0
 -2  -27   37 0
-27  -36   40 0
 18   39  -40 0
 -2  -19   31 0
  5   18  -30 0
-31  -43  -50 0
 10  -30   43 0
 10  -41   43 0
 19   21   29 0
 37   42   45 0
-20   27   40 0
-21  -36   48 0
 31  -36  -48 0
  3   -9  -18 0
 16  -40  -47 0
  1  -18   21 0
  2   28   32 0
 -1  -24  -50 0
-12   35   49 0
 -6  -36   45 0
  7   12  -43 0
  7   30  -43 0
 -5    9  -17 0
  3   14   50 0
-12   17  -49 0
 24   34   49 0
 14  -20   24 0
 -9   35  -49 0
 -4  -47   50 0
  4   44  -44 0
 28  -28  -38 0
  2    4  -48 0
-20   35  -44 0
 30  -31  -43 0
-14  -29   35 0
-20   35  -35 0
 19  -22  -24 0
 25  -28   48 0
-14  -34   44 0
  9   20   44 0
 -3    9  -29 0
 17   34  -34 0
 12   48   48 0
-12  -25  -43 0
-25  -31   48 0
 14  -16   49 0
 -3   -4  -35
```


```
p cnf 50 80
 16   23   42 0
-16   23   42 0
 26   41  -42 0
-26   41  -42 0
```

More Extensions

Resolution: $\dfrac{x \vee C \qquad \neg x \vee C'}{C \vee C'}$

```
 3  2
 1  0
14 13
21 23
20 19 79
12 11
18 16 15 81 78 80
 8  7
 5  6  9 83  4 76
77 84
80 85
17 88 89
15 87 89
18 91 89 90
```

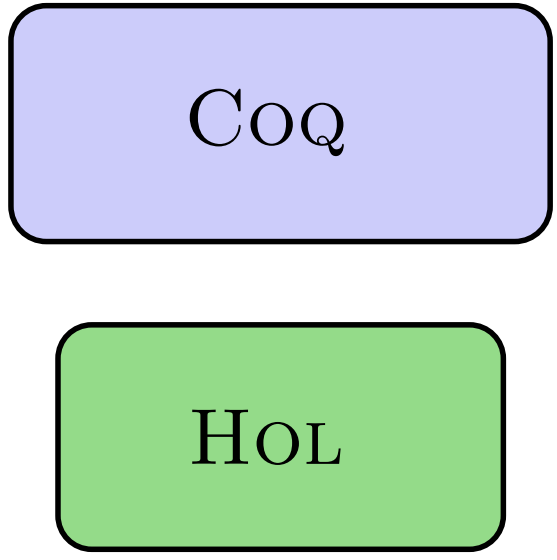
More Extensions

Problem	Vars	Clauses	zChaff	zVerify
dubois50	150	400	0.00	0.01
barrel5	1407	5383	0.50	0.07
barrel6	2306	8931	1.74	0.14
barrel7	3523	13765	5.20	0.26
6pipe	15800	394739	42.21	2.86
longmult14	7176	22390	408.55	7.34
hole11	132	738	14.82	0.90
hole12	156	949	144.49	4.85
hole13	182	1197	5048.23	-

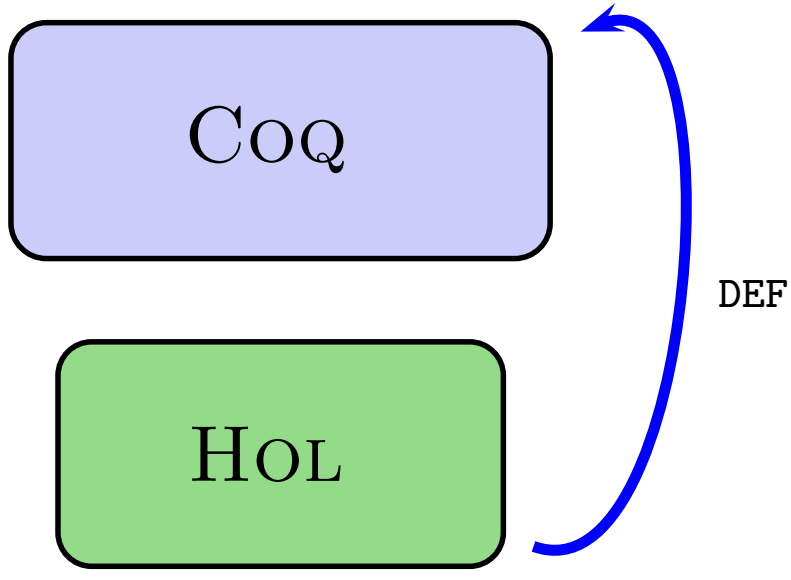
More Extensions

Problem	zVerify	Coq	Cert	Typing	Check
dubois50	0.01	0.04	0.00	0.02	0.02
barrel5	0.07	0.47	0.00	0.32	0.15
barrel6	0.14	1.15	0.08	0.62	0.45
barrel7	0.26	1.45	0.17	0.80	0.48
6pipe	2.86	24.73	0.98	13.92	9.83
longmult14	7.34	73.63	7.72	27.07	38.84
hole11	0.90	0.41	2.96	6.14	1.39
hole12	4.85	58.28	2.44	18.47	37.38
hole13	-	1068.30	88.15	387.44	592.72

More Extensions



More Extensions



More Extensions

	Number		Time			Memory	
Bench.	Theorems	Lemmas	Rec.	Exp.	Comp.	H.D.D.	Virt. Coq
Stdlib	1,726	195,317	2 min 30	6 min 30	10h	218 Mb	4.5 Gb
Model	2,121	322,428	6 min 30	29 min	44h	372 Mb	7.6 Gb
Vectors	2,606	338,087	6 min 30	21 min	39h	329 Mb	7.5 Gb

Conclusions

Conclusions

Key Role of Certificate

Conclusions

Key Role of Certificate

Granularity

Conclusions

Key Role of Certificate

Granularity

Clear Separation

Conclusions

Key Role of Certificate

Granularity

Clear Separation

Room for Improvement