Certification automatique de preuves de terminaison

E. Contejean[1] P. Courtieu[2] J. Forest[2] O. Pons[2]
X. Urbain[2]

Projet A3PAT (ANR)

[1]LRI, Université Paris-Sud
[2]CEDRIC, CNAM

A3PAT

- Démonstrateurs automatiques de terminaison complexes $\Rightarrow$ crédibilité des preuves ?

- Certifier les preuves dans un assistant à la démonstration (sûr).
- Cime $\rightsquigarrow$ traces complètes vers Coq :

```
Inductive R : term → term → Prop := ...
Lemma well_founded_R : well_founded R.
```

Exemple de système

$$R : \begin{cases} (\#)0 & \to \# \\ \# + x & \to x \\ x + \# & \to x \\ (x)0 + (y)0 & \to (x + y)0 \\ (x)0 + (y)1 & \to (x + y)1 \\ (x)1 + (y)0 & \to (x + y)1 \\ (x)1 + (y)1 & \to (x + y + (\#)1)0 \end{cases}$$

Exemple de preuve (CiME)

```
Entering the termination expert.
The dependency graph is (6 nodes)
Checking each of the 1 strongly connected
components :
Checking component 1
Trying simple graph criterion.
Trying to solve the following constraints :
(12 termination constraints)
Search parameters : linear polynomials,
coefficient bound is 3.
Solution found for these constraints :
< polynômes >
```

Nécessité de comprendre la technique pour croire/vérifier la preuve.

Un assistant à la démonstration : Coq

- CIC : syntaxe pour les types, les termes, les propriétés et les preuves.
- Validité des preuves = Vérification des types.
- Exemples :
  - $\lambda x : A.x \ : \ A \to A$
  - $\lambda f : A \to B.\lambda x : A.(f\ x) \ : \ (A \to B) \to A \to B$
- Types inductifs et relations inductives, principes d'induction.

Un assistant à la démonstration : Coq

- CIC : syntaxe pour les types, les termes, les propriétés et les preuves.
- Validité des preuves = Vérification des types.
- Exemples :
  - $\lambda x : A.x \ : \ A \to A$
  - $\lambda f : A \to B.\lambda x : A.(f\ x) \ : \ (A \to B) \to A \to B$
- Types inductifs et relations inductives, principes d'induction.

- Code de coq $\sim$ 160 000 lignes.
- Seul le noyau est critique (qq milliers de lignes).
- Pas convaincus : Coq produit un λ-terme de preuve $\Rightarrow$ possibilité de récrire le vérificateur de types.

Définir les TRS dans un assistant à la démonstration

Deux solutions classiques :

- Shallow embedding,
- Deep embedding.

Shallow embedding

- Algèbre $\Longleftrightarrow$ type inductif,
- TRS $\Longleftrightarrow$ relation inductive.

Avantages :

facile à implanter, pas de ”meta notions”.

Inconvénients :

Impossible de certifier les techniques de preuve

$\Rightarrow$ Chaque critère prouvé sur chaque système

$\Rightarrow$ Termes de preuve complexes.

Deep embedding

- Notions génériques de signatures, termes, substitutions, ...
- Utilisation de bibliothèques (Coccinelle, Color, ...).

Avantages :

Utilisation de théorèmes généraux de correction des critères
$\Rightarrow$ Termes de preuve souvent plus simples.

Inconvénients :

Preuves des théorèmes complexes,
Gain en taille parfois non évident.

Solution intermédiaire

Actuellement,

- DP (et critère de graphes) $\rightsquigarrow$ shallow.
- RPO + AFS $\rightsquigarrow$ deep.
- Interprétations polynomiales $\rightsquigarrow$ shallow.

En fait :

- Prémisses faciles à vérifier $\rightsquigarrow$ deep : RPO . . .
- Prémisses complexes à vérifier $\rightsquigarrow$ shallow : graphe. . .

Premier exemple : Interprétations polynomiales

```
let F_add = signature "
  # : constant ; 0,1 : postfix unary ;
  + : infix binary ; ";

0 + 1;
```

Cime

Coq

```
(* Term algebra *)
Inductive term : Set :=
  | sig_sharp : term
  | sig_0 : term → term
  | sig_1 : term → term
  | sig_plus : term → term → term.

(sig_plus sig_0 sig_1)
```

Système en CiME

```
let X = vars "x y z";

let R_add = TRS F_add X "
      (#)0 → #;
      # + x → x;
      x + # → x;
      (x)0 + (y)0 → (x+y)0;
      (x)0 + (y)1 → (x+y)1;
      (x)1 + (y)0 → (x+y)1;
      (x)1 + (y)1 → (x+y+(#)1)0;
";
```

Schéma de la preuve

```
Inductive R : term → term → Prop :=
(* règles *)
(* (#)0 → #; *)
 | R_rule_0 : (sig_0 sig_sharp) -[R]> sig_sharp
 | ...
(* passage au contexte *)
 | R_sig_0_0 : ∀ x_0 y:term, x_0 -[R]> y →
    (sig_0 x_0) -[R]> (sig_0 y)
 | ...

(* Lemmes pour les différents critères *)

Lemma well_founded_R : well_founded R.
Proof.
  (* Application immédiate des lemmes *)
Qed.
```

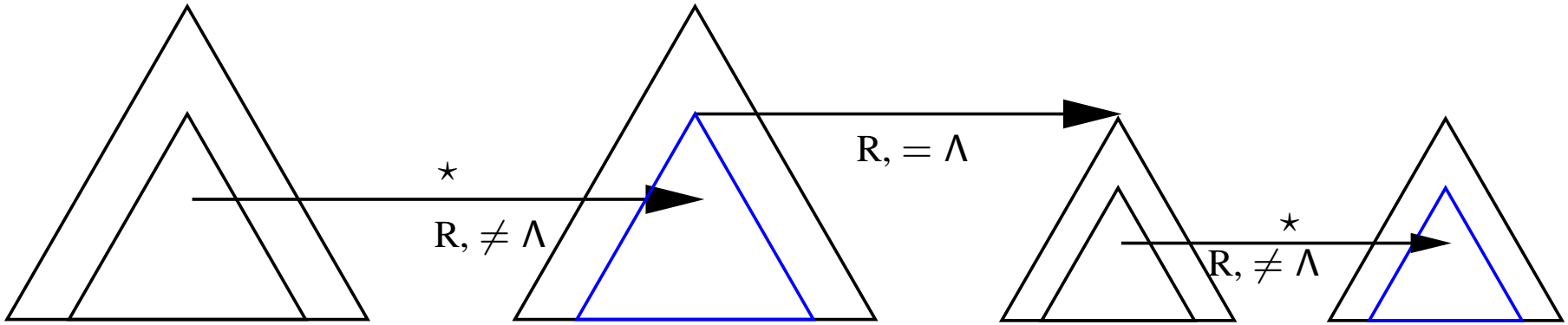
Preuve de termination (CiME)

```
polyinterpkind{("linear",3)};
termcrit "dp";
termination R_add;

  ↪ paires de dépendance:

    1: < (x)1 + (y)1 , x + y >
    2: < (x)1 + (y)1 , (x + y) + (#)1 >
    3: < (x)1 + (y)1 , ((x + y) + (#)1)0 >
    4: < (x)1 + (y)0 , x + y >
    5: < (x)0 + (y)1 , x + y >
    6: < (x)0 + (y)0 , x + y >
    7: < (x)0 + (y)0 , (x + y)0 >
```

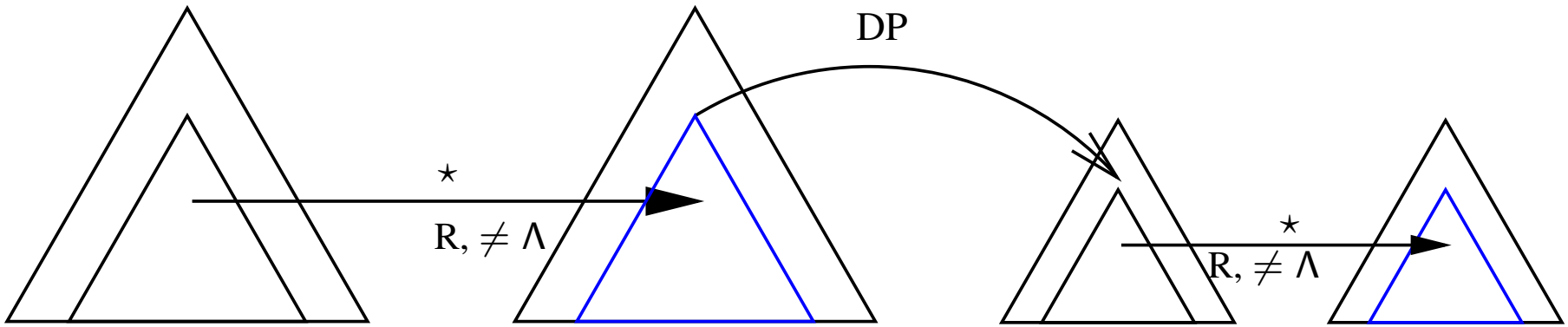
Paires et chaîne de dépendance




```
Inductive DPR : term → term → Prop :=

  (* < (x)0 + (y)0 , (x + y)0 > *)
  | DPR_0 : ∀ x_0 x_1 V_0 V_1,
    x_0 -[R]*> (sig_0 V_0)
  → x_1 -[R]*> (sig_0 V_1)
  → (sig_plus x_0 x_1) -[DPR]> (sig_0(sig_plus V_0 V_1))
  | ...
```

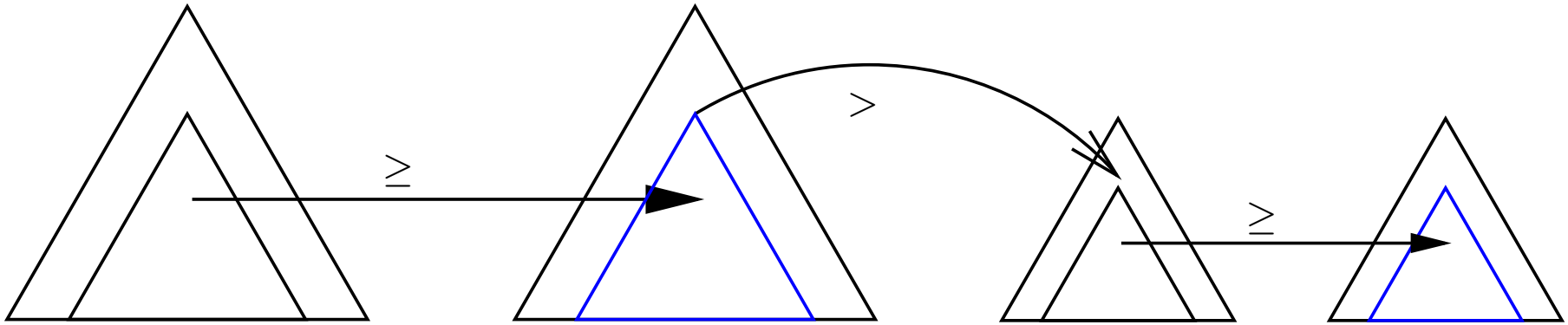
Paires et chaîne de dépendance




```
Inductive DPR : term → term → Prop :=

  (* < (x)0 + (y)0 , (x + y)0 > *)
  | DPR_0 : ∀ x_0 x_1 V_0 V_1,
    x_0 -[R]*> (sig_0 V_0)
  → x_1 -[R]*> (sig_0 V_1)
  → (sig_plus x_0 x_1) -[DPR]> (sig_0(sig_plus V_0 V_1))
   | ...
```

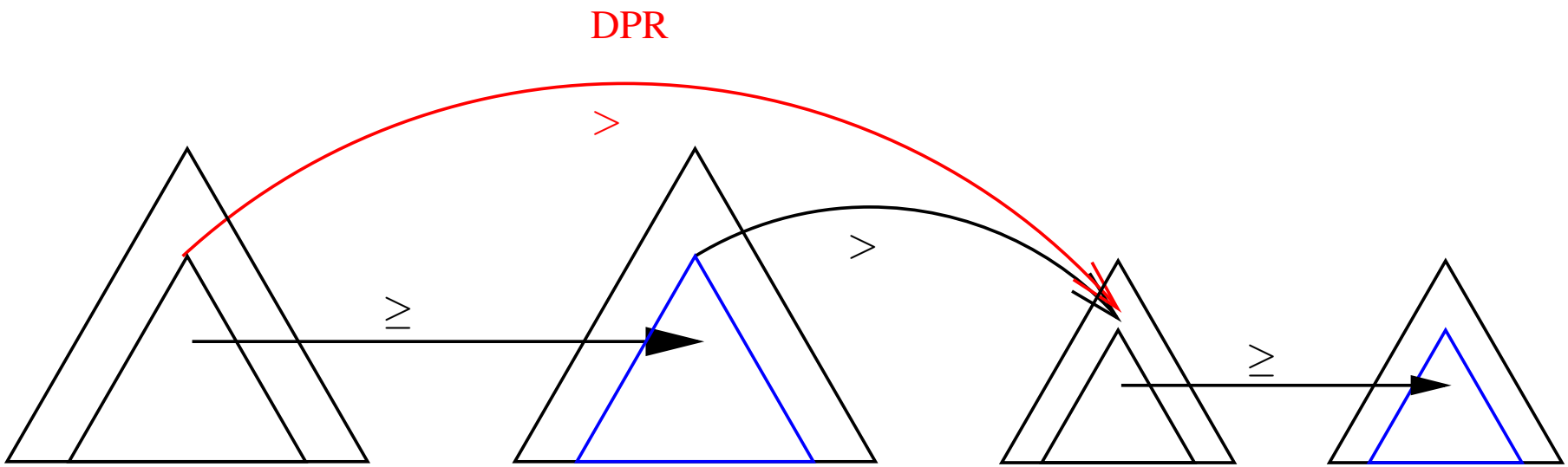
Paires et chaîne de dépendance




```
Inductive DPR : term → term → Prop :=

  (* < (x)0 + (y)0 , (x + y)0 > *)
  | DPR_0 : ∀ x_0 x_1 V_0 V_1,
    x_0 -[R]*> (sig_0 V_0)
  → x_1 -[R]*> (sig_0 V_1)
  → (sig_plus x_0 x_1) -[DPR]> (sig_0(sig_plus V_0 V_1))
   | ...
```

Paires et chaîne de dépendance




```
Inductive DPR : term → term → Prop :=

  (* < (x)0 + (y)0 , (x + y)0 > *)
  | DPR_0 : ∀ x_0 x_1 V_0 V_1,
    x_0 -[R]*> (sig_0 V_0)
  → x_1 -[R]*> (sig_0 V_1)
  → (sig_plus x_0 x_1) -[DPR]> (sig_0(sig_plus V_0 V_1))
   | ...
```

Critère de paire de dépendance

```
Lemma wfR_wfDPR: well_founded DPR → well_founded R.
Proof.
  (* Preuve du critère sur R *)
Qed.
```

⇒ 2^{e} partie : prouver `well_founded DPR`.

Interprétation

```
  Solution found for these constraints:
  [#] = 0;
  [0](X0) = X0 + 1;
  [1](X0) = X0 + 2;
  [+](X0,X1) = X1 + X0;
```

```
Fixpoint measureDPR (t:term) {struct t} : Z :=
 match t with
  | (sig_sharp) ⇒ 0
  | (sig_0 x0) ⇒ measureDPR x0 + 1
  | (sig_1 x0) ⇒ measureDPR x0 + 2
  | (sig_plus x0 x1) ⇒ measureDPR x1 + measureDPR x0
end.
```

Interprétation

```
Lemma measure_DPR_sub_0_pos : ∀ t, 0<= measure_DPR_sub_0 t.
Proof.
(* Induction sur t + ''omega'' *)
Qed.
Lemma DecreaseR : ∀ x y : term ,
                  x -[R]> y → measureDPR x <= measureDPR y
Proof.
  (* Analyse de cas sur R +  ''omega'' *)
Qed.
Lemma wf_DPR : well_founded DPR.
Proof.
  (* Bonne fondation de < sur >= 0
  + DecreaseR + ''omega'' *)
Qed.
Lemma well_founded_R : well_founded R.
Proof.
  exact (wfR_wfDPR wf_DPR).
Qed.
```

Deep embedding

- Type de module `decidable_set` : formalisation des ensembles avec égalité décidable.
- Type de module `Signature` :ensemble décidable + fonction d'arité
- Type de module `Term` contenant :
  - Signature,
  - Ensemble decidable de variables,
  - Définition inductive des `term`,
  - Définition des substitutions,
  - Notions diverses (termes bien formés, etc.)

Deep embedding

```
Module Make (F : Signature) (V : decidable_set.S) :
    (Term F V).
  (* Définitions et preuves de toutes les propriétés du
  module type Term *)
  ...
End Make.
```

- Propriétés vérifiées à la compilation.
- À l'instantiation de `Make`, propriétés vérifiées par construction.

Deuxième exemple : DP + RPO/poly + graphe

DÉMO

Conclusions et pespectives

Techniques actuellement certifiables

- Paires de dépendance,
- Graphes,
- Marques,
- Rpo + AFS,
- Interprétations polynomiales.

Perspectives

- Autres assistants (Isabelle, PhoX, ...).
- Langage de trace intermédiaire CiME / assistants.
- Langage de trace universel.
- Utilisation de démonstrateur pour les preuves de terminaison venant des assistants ?

A3PAT (ANR)

Objectifs :

Certificats de propriétés complexes liées à la récriture.

- Définition d'un langage de traces suffisamment expressif,
- Formalisation et modélisation des notions en jeu comme les paires critiques, unification, terminaison...

Partenaires et membres :

- Cedric : X. Urbain, P. Courtieu, D. Delahaye, C. Dubois, J. Forest et O. Pons.
- INRIA Sophia-Antipolis : Y. Bertot.
- LaBRI : P. Castéran.
- LRI-PCRI : M. Biernacka, S. Conchon et E. Contejean.

Page web du projet :

`http://www3.iie.cnam.fr/~urbain/a3pat/`

Démonstrateurs automatiques de terminaison

- Paires de dépendances :

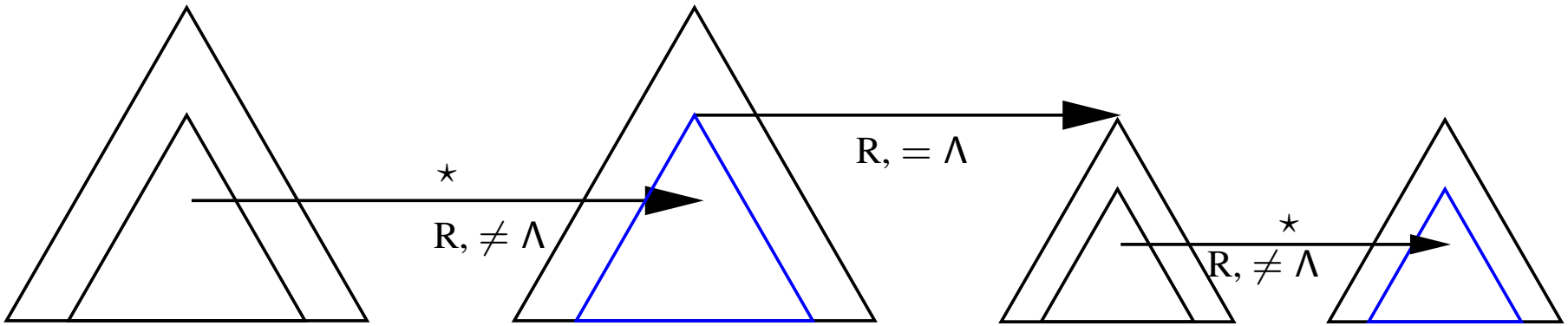
Graphes :

- Graphe $\mathcal{G}$ d'appel des paires de dépendance.
- Un ordre par cycle $\mathcal{C}$ de $\mathcal{G}$ tel que :

$$t \longrightarrow s \Rightarrow s \trianglelefteq t, < l, r > \in \mathcal{C} \Rightarrow r <\!\!= l \text{ et } \exists < l, r > \in \mathcal{C}, r < l$$

Démonstrateurs automatiques de terminaison

- Paires de dépendances :



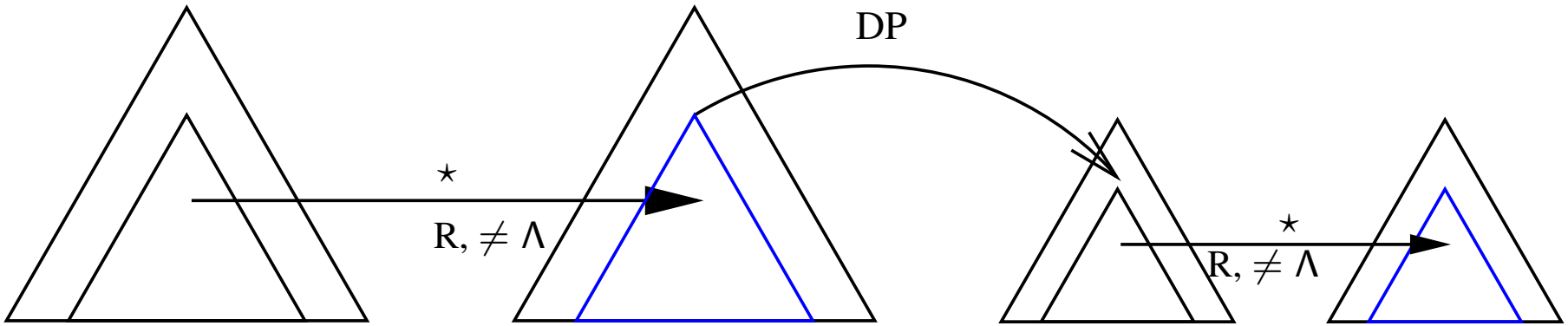
Graphes :

- Graphe $\mathcal{G}$ d'appel des paires de dépendance.
- Un ordre par cycle $\mathcal{C}$ de $\mathcal{G}$ tel que :

$$t \longrightarrow s \Rightarrow s \leq t, < l, r > \in \mathcal{C} \Rightarrow r <= l \text{ et } \exists < l, r > \in \mathcal{C}, r < l$$

Démonstrateurs automatiques de terminaison

- Paires de dépendances :



Graphes :

- Graphe $\mathcal{G}$ d'appel des paires de dépendance.
- Un ordre par cycle $\mathcal{C}$ de $\mathcal{G}$ tel que :

$$t \longrightarrow s \Rightarrow s \leq t, < l, r > \in \mathcal{C} \Rightarrow r <= l \text{ et } \exists < l, r > \in \mathcal{C}, r < l$$

Démonstrateurs automatiques de terminaison

- Paires de dépendances :
  - $< l, r >$ si $(l, rhs) \in \mathcal{R}$, $r \blacktriangleleft rhs$ et $head(r)$ défini.
  - Ordre tel que $t \longrightarrow s \Rightarrow s \leq t$ et $< l, r > \Rightarrow r < l$.

Graphes :

- Graphe $\mathcal{G}$ d'appel des paires de dépendance.
- Un ordre par cycle $\mathcal{C}$ de $\mathcal{G}$ tel que :

$$t \longrightarrow s \Rightarrow s \leq t, < l, r > \in \mathcal{C} \Rightarrow r <= l \text{ et } \exists < l, r > \in \mathcal{C}, r < l$$

Démonstrateurs automatiques de terminaison

- Paires de dépendances :
  - $< l, r >$ si $(l, rhs) \in \mathcal{R}$, $r \blacktriangleleft rhs$ et $head(r)$ défini.
  - Ordre tel que $t \longrightarrow s \Rightarrow s \leq t$ et $< l, r > \Rightarrow r < l$.

Graphes :

- Graphe $\mathcal{G}$ d'appel des paires de dépendance.
- Un ordre par cycle $\mathcal{C}$ de $\mathcal{G}$ tel que :

$$t \longrightarrow s \Rightarrow s \leq t, < l, r > \in \mathcal{C} \Rightarrow r <= l \text{ et } \exists < l, r > \in \mathcal{C}, r < l$$