Problèmes critiques et preuves formelles

Micaela Mayero

Université Paris 13,
LIPN UMR 7030-LCR team
http://www-lipn.univ-paris13.fr/~mayero

22 Novembre 2012

Motivations générales

- Début de l'informatique, premiers algorithmes : 1950
- Ordinateurs (machines) pour calculer (1645, 1936)... puis démontrer (1967, 1984)
- Démontrer quoi ?
  - des théorèmes mathématiques
  - des programmes/logiciels
- Pourquoi ?
  - rigueur
  - aléas des tests, ...
- Comment ?
  - spécification $\rightarrow$ formalisation $\rightarrow$ preuves (suite de déductions logiques vérifiées automatiquement)
  - annotations (logique de Hoare) $\rightarrow$ obligations de preuves $\rightarrow$ preuves (automatiques ou pas)
- interactions : apports réciproques

Résumé des travaux présentés

- Analyse numérique
- Réseaux de Petri
- Approximation Polynomiale Rigoureuse
- Calcul symbolique : SCHUR
- Les nombres réels dans Coq

Focus : Les nombres réels dans Coq

- Généralités
- Reals (stdlib)
- CoRN
- Incompatibilités
- Tentatives d'unification

Perspectives : récapitulatif et nouveauté

- Bilan
- Récapitulatif
- Nouveauté

Motivations

[JAR ; 2012] [ITP2010]
ANR 2005 CerPAN, ANR 2008 FOST
(S. Boldo, F. Clément, J.-C. Filliâtre, M. Mayero, G ; Melquiond, P. Weis)

- Histoire
  - 1997 : premiers contacts avec des numériciens
  - 2005 : CerPAN
  - 2008 : FOST
  - aujourd'hui : les interactions augmentent...
- Problèmes numériques critiques (systèmes embarqués, calculs de ponts, sismologie, contrôles dans les centrales nucléaires,...)
- Avoir confiance dans ces programmes en utilisant les preuves formelles

Problématique

- Quelles études de cas ?
  - gradient analytique
  - équation des ondes 1D
- Choix des approches
  - convergence du schéma numérique : méthode énergétique, méthode de Fourier
  - quels nombres réels, extraction ?
  - gestion de l'erreur liées aux nombres flottants et de l'erreur de méthode
- Spécification : le "grand O" uniforme (deux variables)

▸ détails

Résultats et perspectives

- Interaction entre l'analyse numérique et les preuves formelles
- Une preuve formelle de convergence (et erreurs d'arrondi) pour l'équation des ondes 1D
- Lien avec les preuves sur les erreurs de flottant

↪ Passer à un "vrai" problème d'analyse numérique (éléments finis)

↪ Outils pour les numériciens (utilisation simple des preuves formelles)

Motivations

[JNASA ; 201O] [NFM2009] [REFINE2008]
(C. Choppy, M. Mayero, L. Petrucci, K. Klai) (N. Ghozlane, Y. Yu, N. Aber)

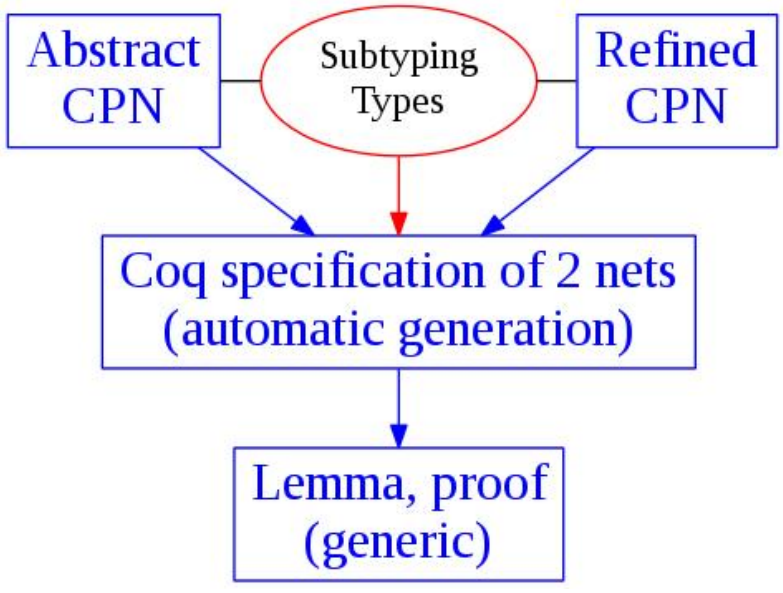
- Réseaux de Petri : formalisme pour modéliser et valider des systèmes critiques : protocoles, synchronisation, systèmes à états, ...
- Raffinement : obtenir une spécification plus concrète (et gérer les explosion d'états).
- Prouver formellement et automatiquement la correction d'un raffinement (préservation des propriétés).
- Utiliser les preuves formelles dans le domaine de la vérification.

Problématique

- Choix de formalisation : 3 expérimentations (Sets, FSets, List)
  - décisif pour les preuves
  - contexte d'évolution des bibliothèques de Coq
  - ajout des couleurs
- Automatisation des preuves de correction
- Gestion des échecs (fail) de preuves

Details

Résultats et perspectives possibles



- Preuve que c'est bien un raffinement (ou pourquoi non)

↪ Génération automatique du code Coq

↪ Propriétés dynamiques (vivacité, blocage) des CPNs

Motivations

[NFM2012]
ANR 2010 TaMaDi
(N. Brisebarre, M. Joldes, E. Martin-Dorel, M. Mayero, J.-M. Muller, I. Paşca, L. Rideau, L. Théry)

RPA :

- Une solution pour représenter des fonctions réelles en machine.
- La seule solution si seulement $\times$, $-$, $+$
- Utilisée dans les systèmes de calcul formel
- (P, Δ) où P polynôme, Δ intervalle contenant l'erreur d'approximation

Contrôler complètement l'erreur d'approximation.
$\hookrightarrow$ Preuves formelles : faisabilité et efficacité des calculs (de Coq) avec des polynômes de Taylor.

Problématique

- Calculer dans Coq (`compute`, `vm_compute`, `native_compute`)
- Quels bibliothèques ? (réels, flottants, intervalles)
- Modularité par rapport à la base de polynômes : poly. de Taylor, de Chebyshev
- Modularité par rapport aux types de données : listes, tableaux
- Modularité : comment ? (types classes, canonical structures, modules)
- Des preuves génériques du prédicat de validité (erreur d'approximation non sous-estimée)
- Fonctions D-finies (holonomes : sol. d'une eq. diff. ord. lin. (LODE) avec coeff. poly.) : oui. Les autres : plus difficile.

Résultats et perspectives

- Développement modulaire
- Calculs efficaces : en moyenne seulement 10 fois plus lents que Sollya (outil dédié aux approx. polynomiales ; écrit en C ; MPFI)
- Preuves achevées pour *inv*, *exp*, *sin*, *cos*, *add*, *comp*, *div*

↪ Fonctions manquantes : *tan* ? (non holonome), *atan*, *ln*, *asin*, *acos*

↪ Améliorer efficacité : Karatsuba (mult), sums_of_squares (positivité des poly.)

↪ Polynômes de Chebyshev

↪ Autres techniques : DDMF (Dynamic Dictionary of Math. Functions) pour générer les récurrences, ... (M. Mezzaroba)

Motivations

[Calculemus2010]
(F. Butelle, F. Hivert, M. Mayero, F. Toumazet)

- SCHUR : outil interactif de combinatoire algébrique (fonctions symétriques, groupes de Lie)
- Explosions combinatoires
- Calculer, énoncer des conjectures, enseignement
- 40 000 lignes de C non commenté et astucieux (traduit de Pascal automatiquement)

Avoir confiance dans ce code critique.
↪ Le prouver formellement (preuve de programme).

Problématique

- On ne peut pas prouver les 40 000 lignes !
- Choisir une fonction
  - significative (100 fois)
  - intéressante (critique ?)
  - pas trop simple
  - pas trop compliquée
- Étude de faisabilité

↪ Le calcul du conjugué d'une partition d'entiers

▸ détails

Résultats et perspectives

- Preuve d'une fonction clé (à isoler) de SCHUR
- Interaction entre deux communautés
- Combinatoristes : accroître la confiance dans leur outil
- Preuves formelles : faisabilité (vers un changement d'échelle)

↪ Preuve d'une fonction énumérative : coefficients de Littlewood-Richardson, nombres de Kostkas, ...

↪ Bibliothèque formellement prouvée

Motivation et résultats

- Formaliser les nombres réels dans un prouveur
- Quels nombres réels ? Comment ?
- Plusieurs bibliothèques dans Coq

↪ Une réflexion sur ces bibliothèques
↪ Une tentative d'unification

Pourquoi est-ce un problème ?

- nombres réels vs nombres à virgule flottante
- nombres représentables en machine
- calculables

- propriétés mathématiques : non associativité de l'addition

(1003+ -1000)+7.501 $\rightsquigarrow$ 10.501000000000012
1003+ (-1000+7.501) $\rightsquigarrow$ 10.500999999999764

Mais nous voulons cette propriété pour les preuves

Quels réels ?

- axiomatisation/construction
- Cauchy, Cantor, coupures de Dedekind
- classiques/constructifs
- 1^{er} ordre/2^{d} ordre

- réels exacts
- réels algébriques
- réels non standards

Focus sur Coq

- 1997, 2002, 2012 : Reals stdlib (Mayero-Desmettre-Lelay)
- 2000, 2009, ... : CoRN (Niqui-Spitters...)
- 2000 : Co-Inductifs (Ciaffaglione...)
- 2008 : arithmétique réelle exacte (O'Connor)
- 2011 : nombres réels exacts (Krebbers...)
- 2011 : réels algébriques (Cohen)

Contexte actuel :

- Reals : C. Lelay (projet Coquelicot)
- CoRN : B. Spitters (Univalent Foundation of Mathematics)

Motivations et questions

(avec M. Niqui, 2010)

- Plusieurs bibliothèques de nombres réels
- axiomatisations ou implantations

↪ Confusion des utilisateurs :

- différences entre CoRN et la bibliothèque standard Reals
- laquelle utiliser ?
- unifier les deux ? :
- une partie commune ?
- ...

↪ Plusieurs problèmes théoriques !

Reals

Corps commutatif ordonné archimédien complet

$$
\begin{array}{lcl}
R & : & Set \\
\\
= & : & R \to R \to Prop \\
+ & : & R \to R \to R \\
\times & : & R \to R \to R \\
0 & : & R \\
\mathrm{up} & : & R \to Z
\end{array}
\qquad
\begin{array}{lcl}
< & : & R \to R \to Prop \\
- & : & R \to R \\
\_^{-1} & : & R \to R \\
1 & : & R
\end{array}
$$

Reals : structure algébrique

$$\forall r_1, r_2 : R, \quad r_1 + r_2 = r_2 + r_1$$
$$\forall r_1, r_2, r_3 : R, \quad (r_1 + r_2) + r_3 = r_1 + (r_2 + r_3)$$
$$\forall r : R, \quad r + -r = 0$$
$$\forall r : R, \quad 0 + r = r$$
$$\forall r_1, r_2 : R, \quad r_1 \times r_2 = r_2 \times r_1$$
$$\forall r_1, r_2, r_3 : R, \quad (r_1 \times r_2) \times r_3 = r_1 \times (r_2 \times r_3)$$
$$\forall r : R, \quad r \neq 0 \rightarrow r \times r^{-1} = 1$$
$$\forall r : R, \quad 1 \times r = r$$
$$1 \neq 0$$
$$\forall r_1, r_2, r_3 : R, \quad r_1 \times (r_2 + r_3) = r_1 \times r_2 + r_1 \times r_3$$

Reals : structure d'ordre

$\forall r_1, r_2 : R,\ \{r_1 < r_2\}$ or $\{r_1 = r_2\}$ or $\{r_1 > r_2\}$

$\forall r_1, r_2 : R,\ r_1 < r_2 \to r_2 \not< r_1$

$\forall r_1, r_2, r_3 : R,\ r_1 < r_2 \to r_2 < r_3 \implies r_1 < r_3$

$\forall r, r_1, r_2 : R,\ r_1 < r_2 \to r + r_1 < r + r_2$

$\forall r, r_1, r_2 : R,\ 0 < r \to r_1 < r_2 \to r \times r_1 < r \times r_2$

$\forall r : R,\ r < \mathrm{up}(r) \leq r + 1$

Tout ensemble borné non vide $E \subset R$ admet une borne supérieure

CoRN

$$
\begin{array}{rl}
\# & : \text{prédicat binaire sur } R \\
= & : \text{prédicat binaire sur } R \\
(x, \phi_x) & : R^* \Leftrightarrow x : R \wedge \phi_x : x \# \mathbf{0} \\
< & : \text{prédicat binaire sur } R \\
+ & : R \to R \to R \\
\times & : R \to R \to R \\
0 & : R \\
\lim & : \{\text{Suites de Cauchy}\} \to R
\end{array}
\qquad
\begin{array}{rl}
- & : R \to R \\
(\_^{-1}, \_) & : R^* \to R \\
1 & : R
\end{array}
$$

CoRN

$$\forall x : R,\ \ \neg(x\#x)$$
$$\forall x, y : R,\ \ (x\#y) \to (y\#x)$$
$$\forall x, y : R,\ \ (x\#y) \to \forall z : R,\ \ (x\#z) \text{ ou } (z\#y)$$
$$\forall x, y : R,\ \ \neg(x\#y) \leftrightarrow (x = y)$$

CoRN

$\forall x_1, x_2, y_1, y_2 : R,\ \ x_1 + y_1 \# x_2 + y_2 \rightarrow x_1 \# x_2$ ou $y_1 \# y_2$

$\forall x, y, z : R,\ \ x + (y + z) = (x + y) + z$

$\forall x : R,\ \ x + 0 = x$

$\forall x, y : R,\ \ x + y = y + x$

$\forall x, y : R,\ \ -x\# - y \rightarrow x \# y$

$\forall x : R,\ \ x + (-x) = 0$

$\forall x_1, x_2, y_1, y_2 : R,\ \ x_1 \times y_1 \# x_2 \times y_2 \rightarrow x_1 \# x_2$ ou $y_1 \# y_2$

$\forall x, y, z : R,\ \ x \times (y \times z) = (x \times y) \times z$

$\forall x : R,\ \ x \times 1 = x$

$\forall x, y : R,\ \ x \times y = y \times x$

$\forall x, y, z : R,\ \ x \times (y + z) = (x \times y) + (x \times z)$

$1 \# 0$

$\forall x : R \forall H_x : x \# 0 . (x^{-1}, H_x) \# 0$

$\forall x : R \forall H_x : x \# 0 . x \times (x^{-1}, H_x) = 1$

CoRN

$\forall x_1, x_2, y_1, y_2 : R,\ \ x_1 < y_1 \to x_2 < y_2$ ou $x_1 \# x_2$ ou $y_1 \# y_2$

$\forall x, y, z : R,\ \ x < y \to y < z \to x < z$

$\forall x : R,\ \ \neg x < x$

$\forall x, y : R,\ \ x < y \to \forall z : R,\ \ x + z < y + z$

$\forall x, y : R,\ \ 0 < x \to 0 < y \to 0 < x \times y$

$\forall x, y : R,\ \ x \# y \leftrightarrow x < y$ ou $y < x$

$\forall x : R \exists n : N, x \leq \mathrm{nring}(n)$

$\forall g : \mathrm{CauchySeq}_R, \forall \epsilon > 0\ \exists M \forall m \geq_N M, \lim g - \epsilon < g(m) < \lim g + \epsilon$

$\mathrm{CauchySeq}_R := \{g : N \to R | \forall \epsilon > 0 \exists M \forall m \geq_N M, g(M) - \epsilon < g(m) < g(M) + \epsilon\}$

$\mathrm{nring}(0) := 0\ ,\quad \mathrm{nring}(n+1) := 1 + \mathrm{nring}(n)$

Reals : sorte, ordre, égalité

- Set, Prop
- inv, div : fonctions totales
- ordre total
- égalité de Leibniz
- axiome de complétude

CoRN : Apartness, ordre, égalité

- L'apartness définit l'égalité : $\neg(x\#y) \leftrightarrow (x = y)$.
- L'ordre définit l'apartness : $x\#y \leftrightarrow x < y$ ou $y < x$
- Pour les suites de Cauchy :

$$\alpha\#\beta \Leftrightarrow \exists\epsilon > 0 \exists N, \forall n > N, |\alpha_N - \beta_N| > \epsilon$$

$$\alpha = \beta \Leftrightarrow \forall\epsilon > 0, \exists N, \forall n > N, |\alpha_N - \beta_N| < \epsilon$$

-

$$\# : R \longrightarrow R \longrightarrow \mathit{Type} \qquad <: R \longrightarrow R \longrightarrow \mathit{Type}$$
$$=: R \longrightarrow R \longrightarrow \mathit{Prop} \qquad \leq: R \longrightarrow R \longrightarrow \mathit{Prop}$$

Problèmes pour unifier

Remarque : Reals satisfait les ax. de CoRN (Kaliszyk–O'Connor, 2009)

- $x \neq 0 \implies x \times x^{-1} = 1$ Ne semble pas prouvable en CoRN : $(x \neq 0 \implies x\#0)$
- $<$ dans Type (récemment dans Prop) et $=$ dans Prop
- $\#$ familier pour l'utilisateur ? ?
- CoRN a une hiérarchie algébrique (setoïde, semi-groupe, groupe, anneau, 'corps'...) ?

NB : 'corps' signifie corps constructif

Possibilités d'unification

Au moins trois possibilités raisonnables :

1. Partie commune jusqu'aux anneaux avec égalité :
   Setoïde $\Longrightarrow$ Groupe $\Longrightarrow$ Anneaux $\langle A, =, +, *, -, 0, 1 \rangle$
   - Corps avec l'inégalité stricte et inverse totale
   - 'Corps' avec apartness (ou l'ordre) et inverse partielle
2. Suivre CoRN avec $<$ au lieu de $\#$
   (axiomatisation de Ciaffaglione's dans Type)
3. Suivre CoRN.

Proposition concrète

- Hiérarchie algébrique de CoRN
- $<$ au lieu de $\#$
- Dichotomie au lieu de l'ordre total
- Complétude avec les suites de Cauchy au lieu de la borne sup.
- Une autre condition de bord pour l'axiome sur l'inverse ($x < 0$ ou $x > 0 \rightarrow \ldots$)

Une implantation existe avec les Type Classes... à suivre

Bilan général

"On" progresse mais il reste beaucoup de travail !

Récapitulatif (projets actifs)

- Analyse numérique :
  - Éléments finis (programme C++)
  - Boîte à outils
- Approximation polynomiale rigoureuse :
  - Améliorer l'efficacité des calculs
  - Polynômes de Chebyshev
- Les nombres réels dans Coq
  - Passer aux sétoïdes
  - Se passer de l'apartness ?

Interaction avec la logique linéaire

Formaliser en Coq :

- les réseaux de preuves : méthode graphique pour représenter la logique linéaire et manipuler les règles ▸ exemple
- les graphes de partage : représentations graphiques du λ-calcul optimisant la mise en commun des ressources

ANNEXES

Étude choisie

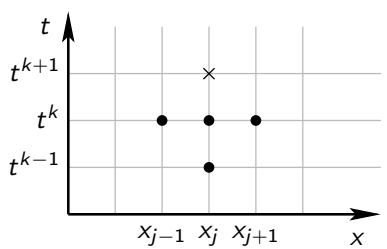
- L'équation des ondes approchée par un schéma numérique discret (grille) :
  On cherche $u : \mathbb{R}^2 \to \mathbb{R}$ suffisamment régulière t.q. :

$$\frac{\partial^2 u(x,t)}{\partial t^2} - c^2 \frac{\partial^2 u(x,t)}{\partial x^2} = s(x,t)$$

  avec $u_0(x)$ la position initiale et $u_1(x)$ la vitesse initiale.

- Un schéma numérique (à trois points) :

u_j^k dépend de u_{j-1}^{k-1}, u_j^{k-1}, u_{j+1}^{k-1} et u_j^{k-2}



Erreur de méthode

Une preuve mathématique de convergence du schéma numérique :
Schéma à trois points : On veut $u_j^k \approx u(j\Delta x, k\Delta t)$.

$$\frac{u_j^k - 2u_j^{k-1} + u_j^{k-2}}{\Delta t^2} - c^2 \frac{u_{j+1}^{k-1} - 2u_j^{k-1} + u_{j-1}^{k-1}}{\Delta x^2} = s_j^{k-1}$$

(idem pour initialiser u_j^0 et u_j^1.)
On vérifie que u et u_j^k sont proches quand $(\Delta x, \Delta t) \to 0$.
On définit $e_j^k \stackrel{\text{def}}{=} \bar{u}_j^k - u_j^k$: l'erreur de convergence
où $\bar{u}_j^k$ est la valeur de u au point (j, k) de la grille.
On veut borner $\left\| e_h^{k_{\Delta t}(t)} \right\|_{\Delta x}$: la moyenne de l'erreur de convergence
en tous les points de la grille au temps $k_{\Delta t}(t) = \left\lfloor \frac{t}{\Delta t} \right\rfloor \Delta t$.
On veut prouver : $\left\| e_h^{k_{\Delta t}(t)} \right\|_{\Delta x} = O_{[0,t_{\max}]}(\Delta x^2 + \Delta t^2)$

Le "grand O"

Grand O usuel $f(\boldsymbol{x}) = O_{\|\boldsymbol{x}\|\to 0}(g(\boldsymbol{x}))$:

$$\exists \alpha, C > 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \quad \|\boldsymbol{x}\| \leq \alpha \Rightarrow |f(\boldsymbol{x})| \leq C \cdot |g(\boldsymbol{x})|$$

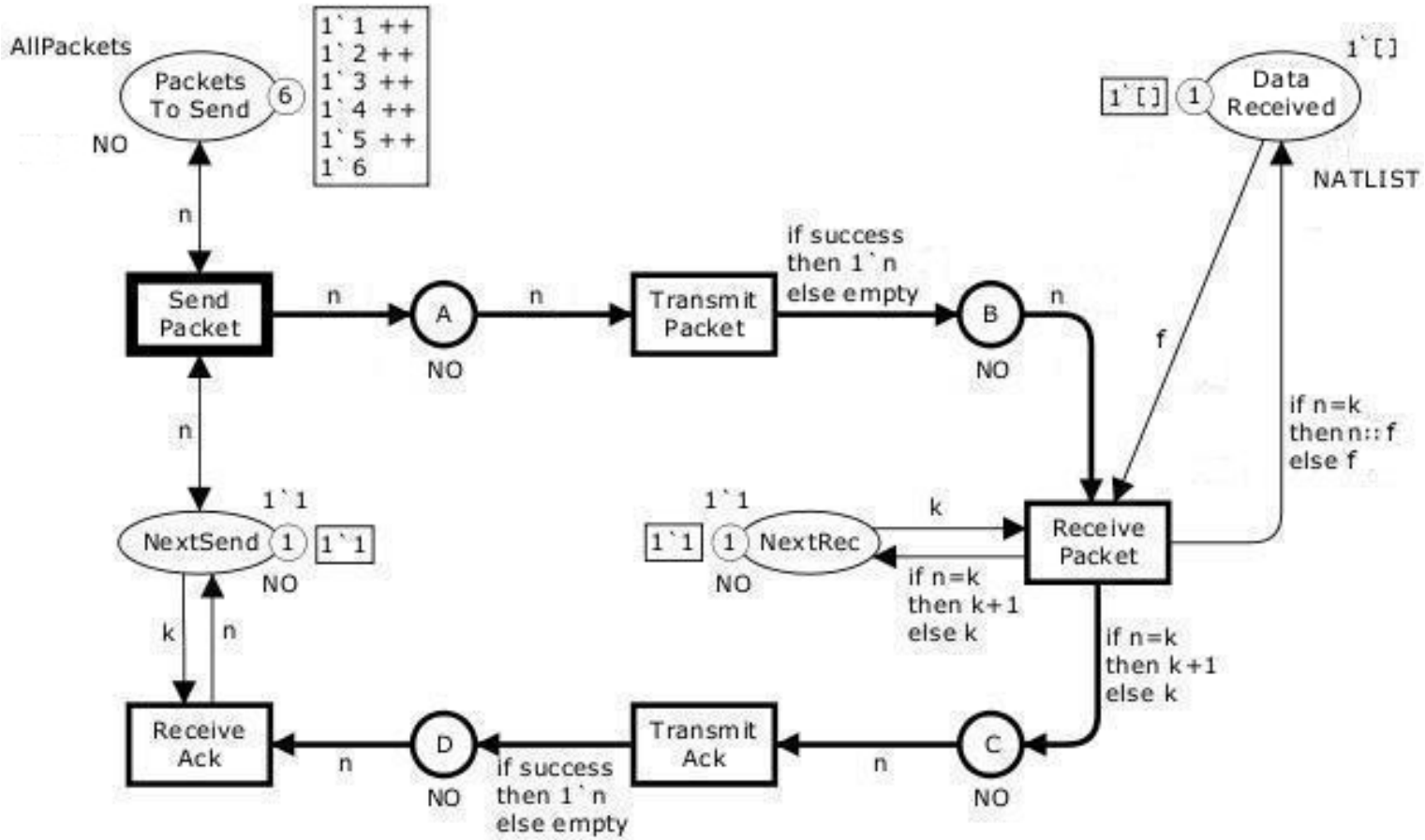
Deux variables : $f(\boldsymbol{x}, \boldsymbol{\Delta x}) = O_{\|\boldsymbol{\Delta x}\|\to 0}(g(\boldsymbol{\Delta x}))$
Si on suppose que l'égalité est vraie pour chaque $\boldsymbol{x} \in \mathbb{R}^2$:

$$\forall \boldsymbol{x}, \exists \alpha > 0, \exists C > 0, \forall \boldsymbol{\Delta x}, \quad \|\boldsymbol{\Delta x}\| \leq \alpha \Rightarrow |f(\boldsymbol{x}, \boldsymbol{\Delta x})| \leq C \cdot |g(\boldsymbol{\Delta x})|$$

(les constantes α et C sont des fonctions de $\boldsymbol{x}$)
$\hookrightarrow$ inutile (borne inf. de α : 0 et la borne sup. de C : $+\infty$)
Grand O uniforme par rapport à $\boldsymbol{x}$, $f(\boldsymbol{x}, \boldsymbol{\Delta x}) = O_{\Omega_{\boldsymbol{x}}, \Omega_{\boldsymbol{\Delta x}}}(g(\boldsymbol{\Delta x}))$:

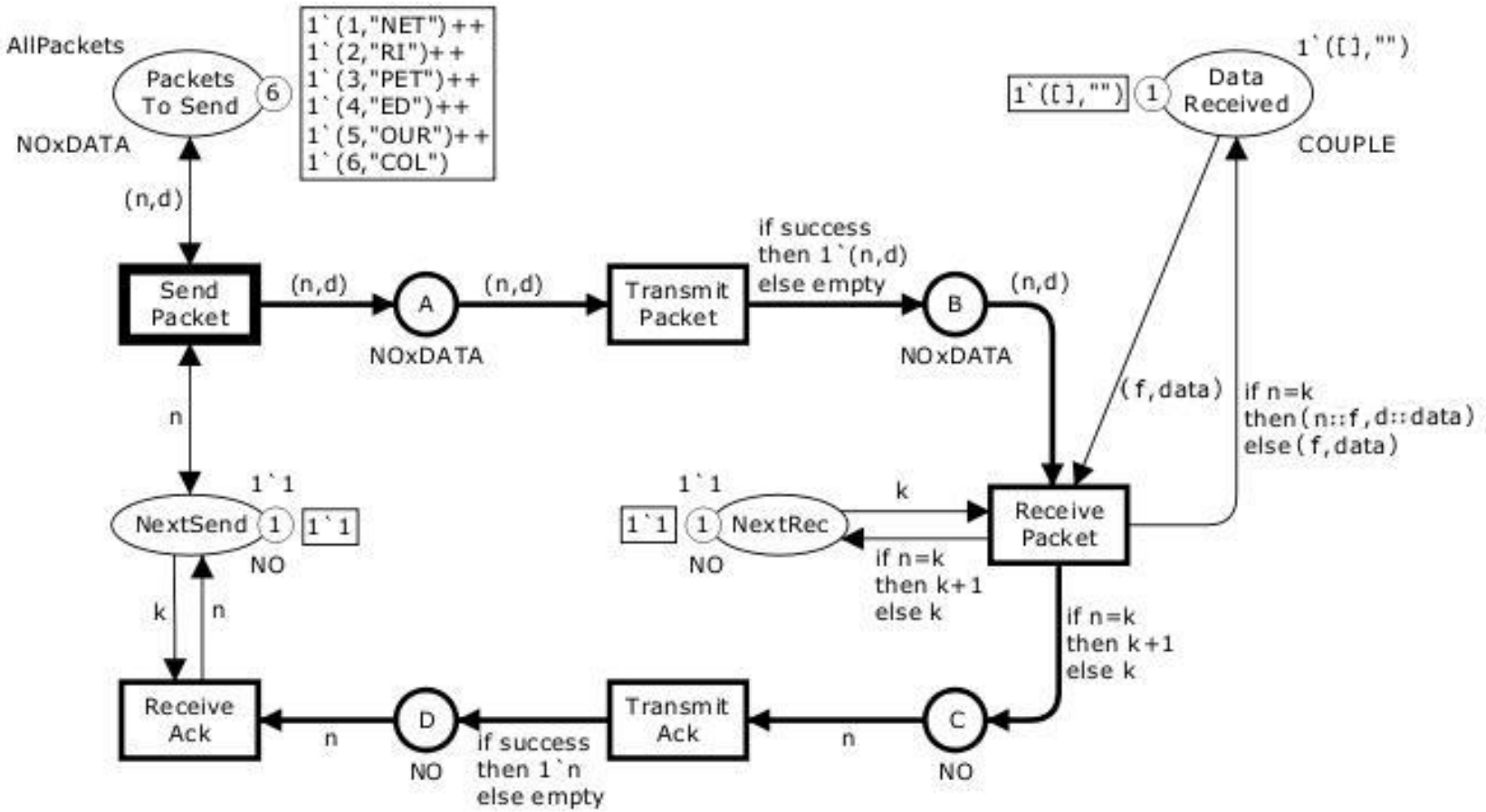
$$\exists \alpha > 0, \exists C > 0, \forall \boldsymbol{x} \in \Omega_{\boldsymbol{x}}, \forall \boldsymbol{\Delta x} \in \Omega_{\boldsymbol{\Delta x}}, \\ \|\boldsymbol{\Delta x}\| \leq \alpha \Rightarrow |f(\boldsymbol{x}, \boldsymbol{\Delta x})| \leq C \cdot |g(\boldsymbol{\Delta x})|$$

Exemple de réseau de Petri coloré (CPN)

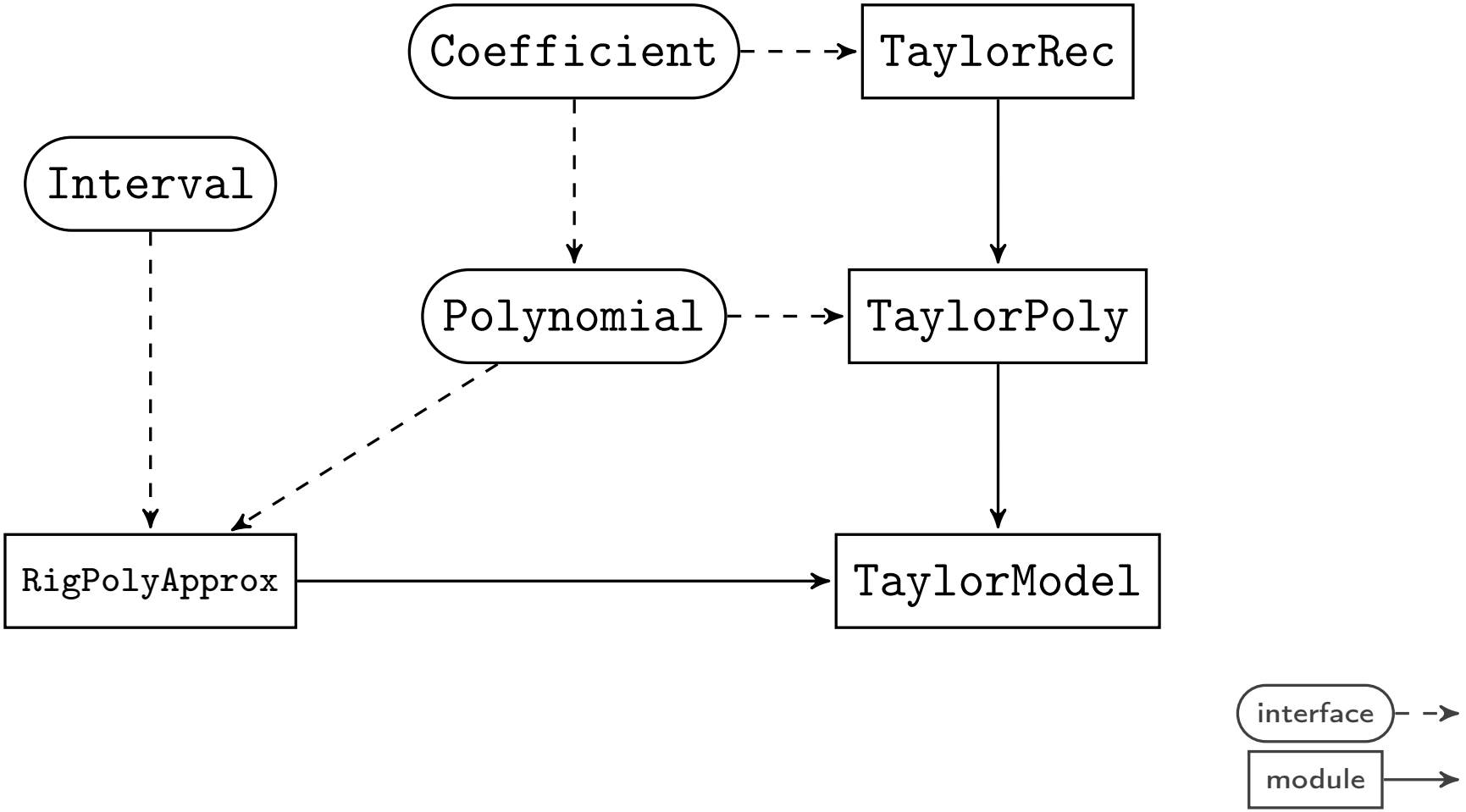


Exemple de raffinement

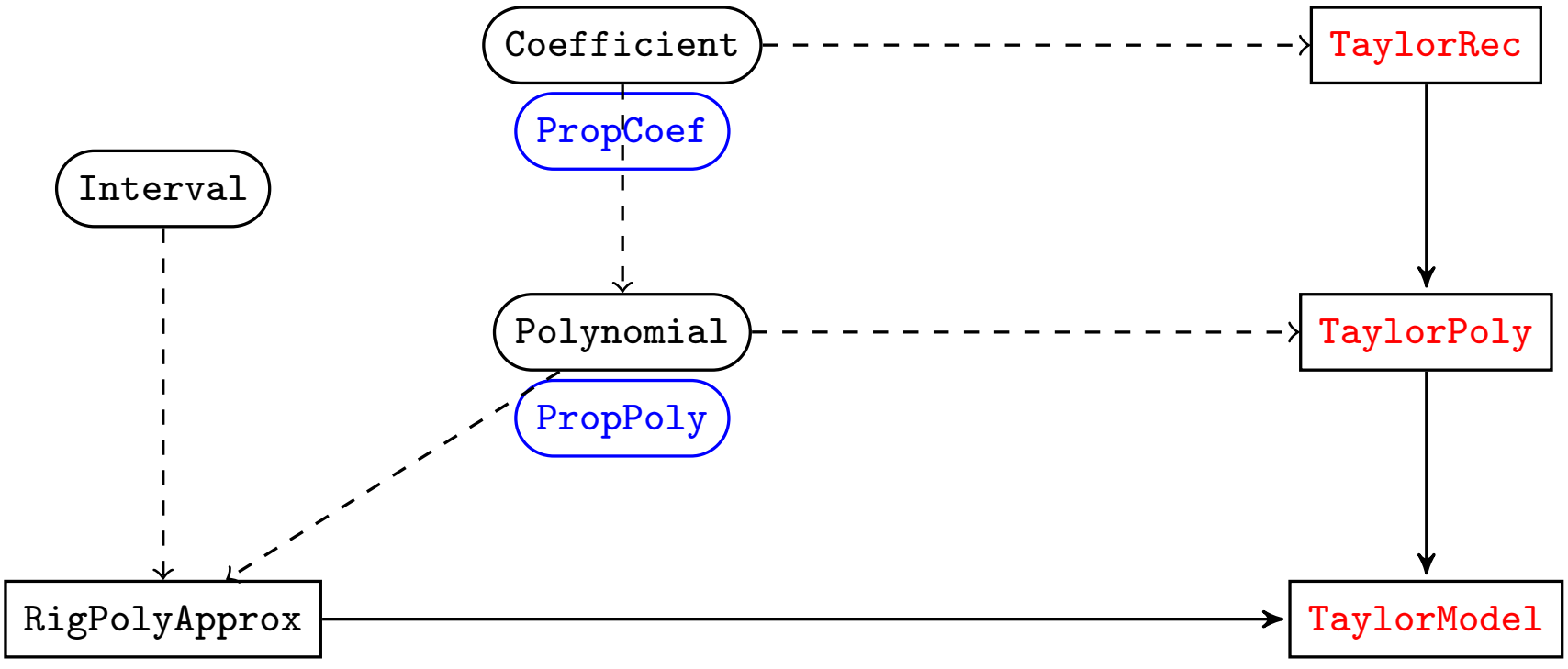
(nœud, sous-réseau, type)



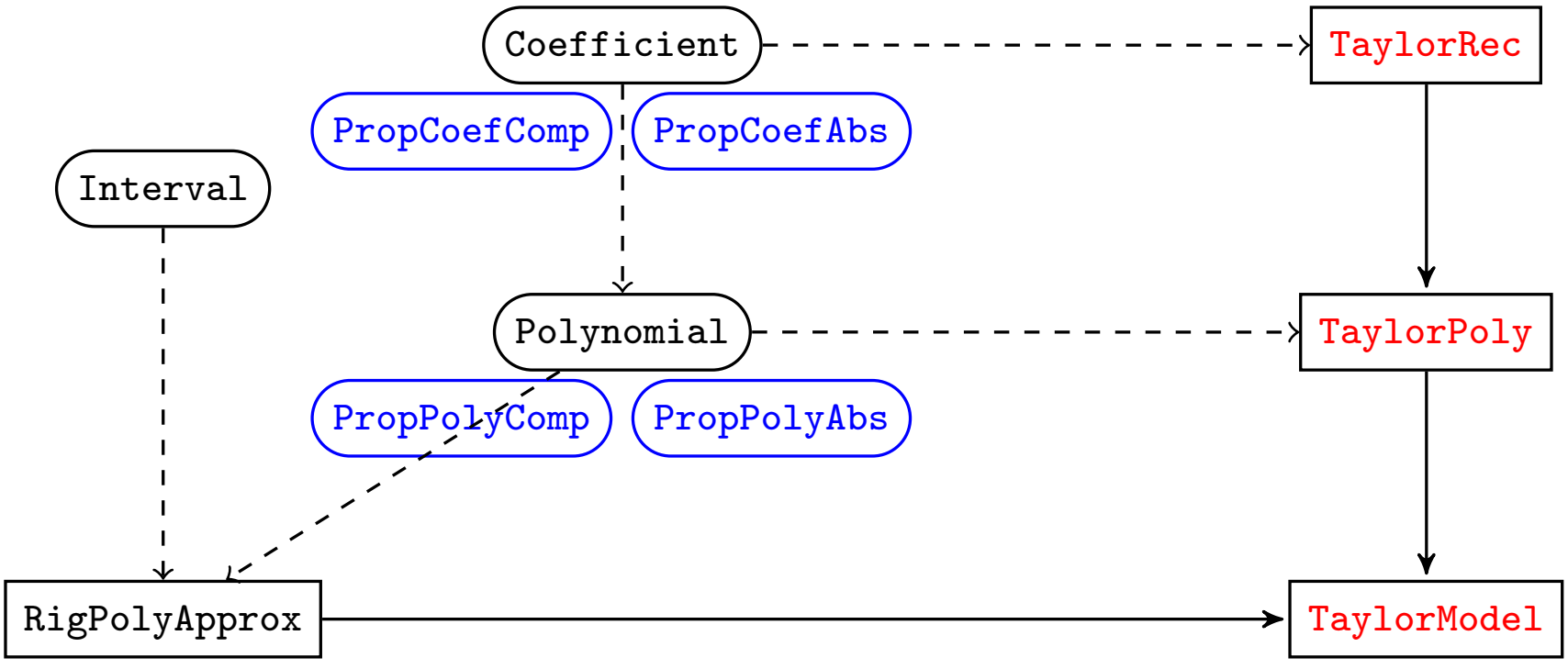
Hiérarchie pour le calcul



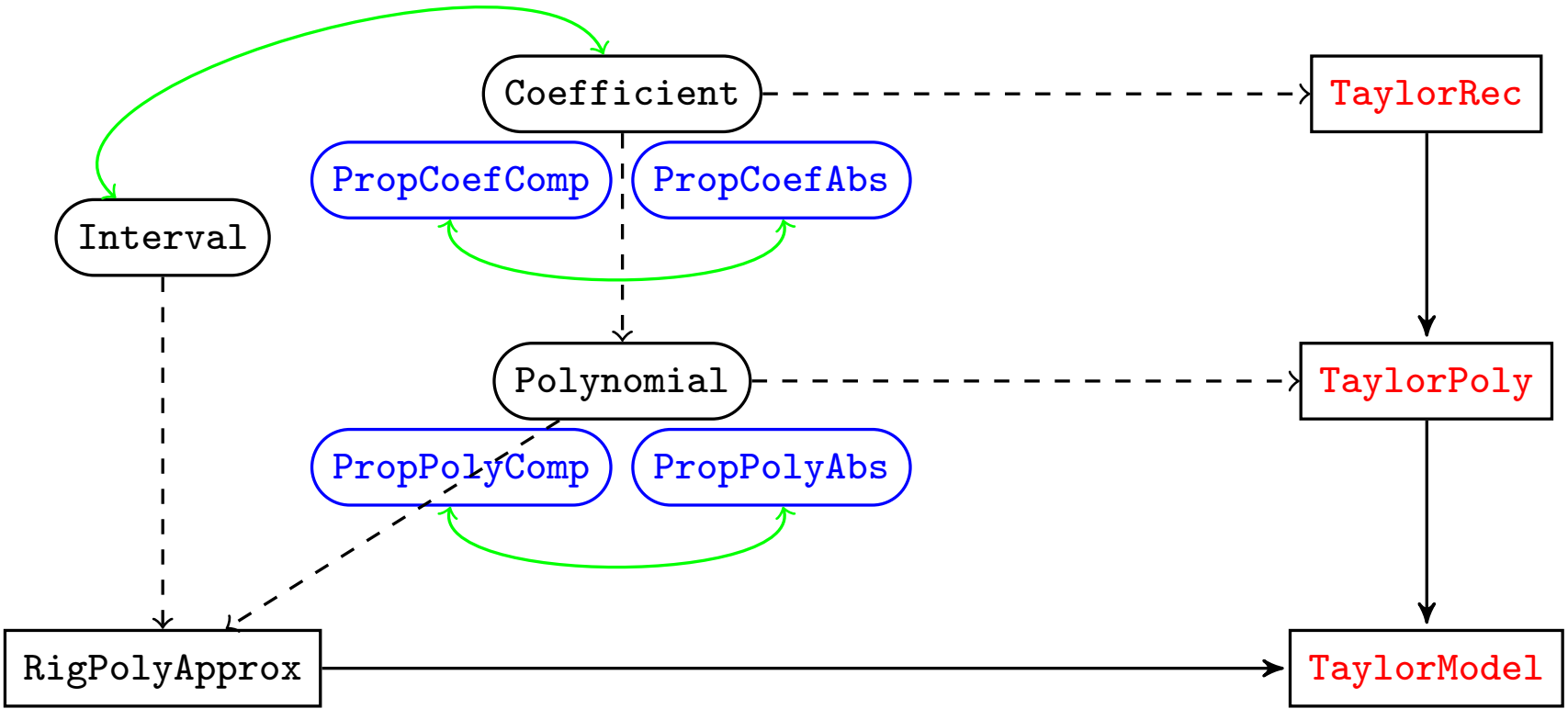
Hiérarchie adaptée aux preuves



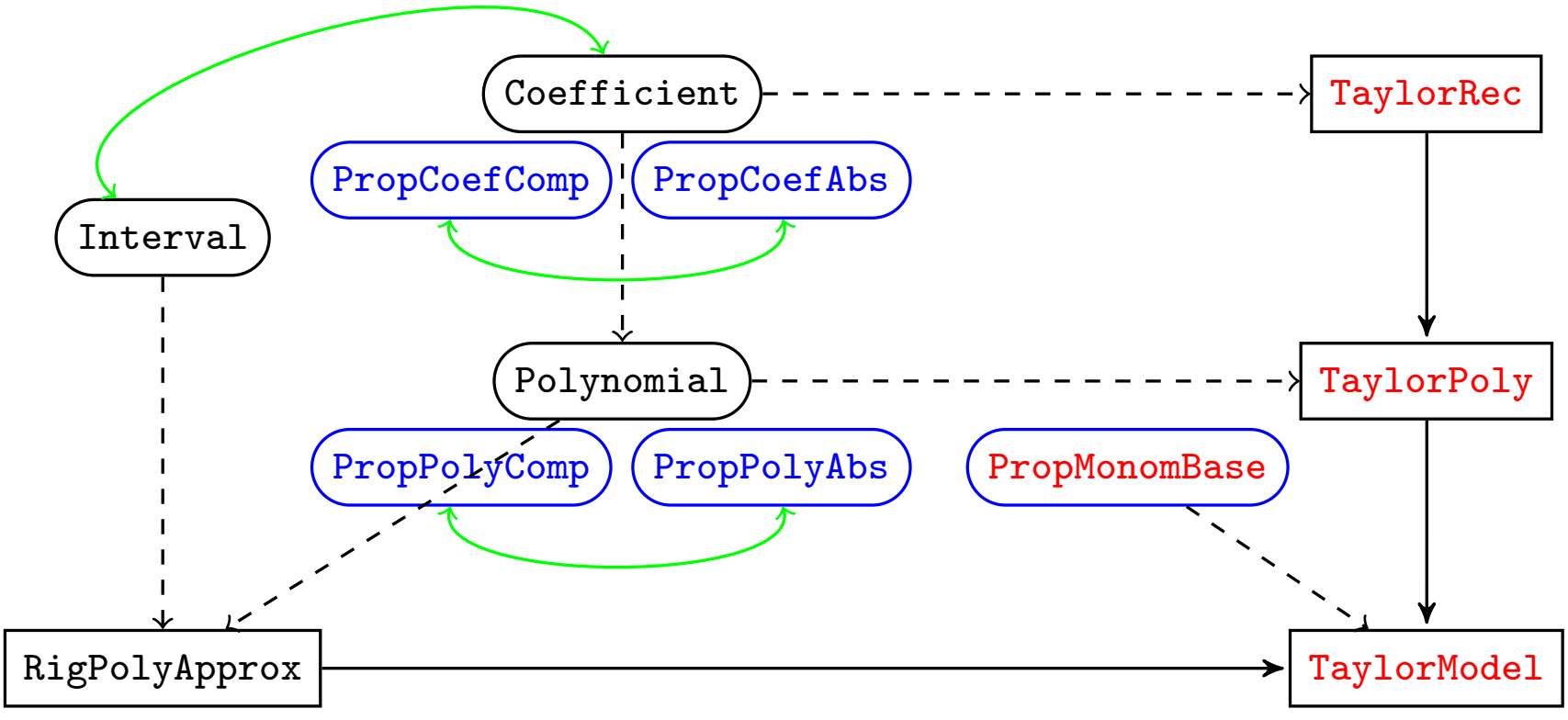
Hiérarchie adaptée aux preuves



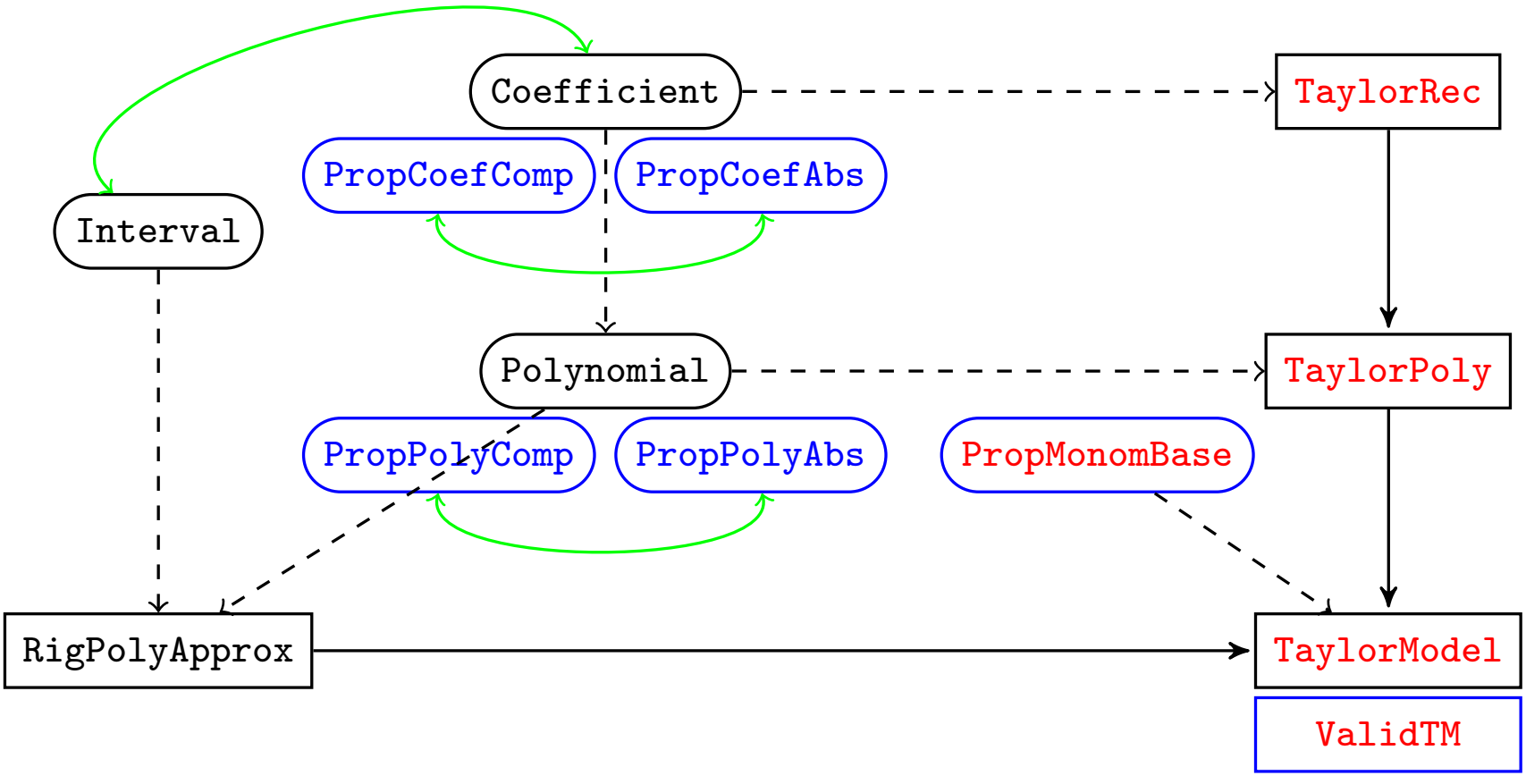
Hiérarchie adaptée aux preuves



Hiérarchie adaptée aux preuves



Hiérarchie adaptée aux preuves

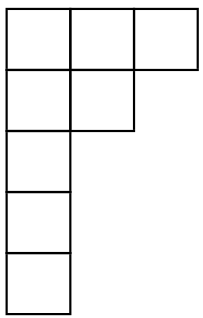


◂ Conclusion

Le conjugué d'une partition d'entiers

Une partition d'un entier n est une façon d'écrire n comme une somme d'une séquence d'entiers positifs non croissants.

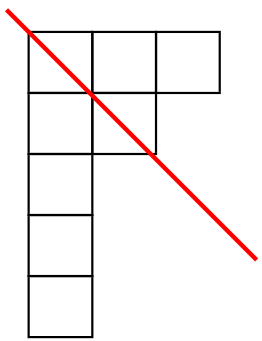
$\lambda = (3, 2, 1, 1, 1)$



Le conjugué d'une partition d'entiers

Une partition d'un entier n est une façon d'écrire n comme une somme d'une séquence d'entiers positifs non croissants.

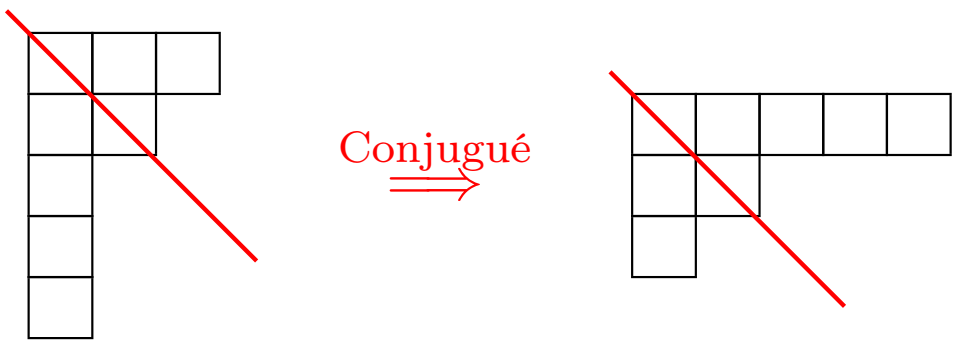
$\lambda = (3, 2, 1, 1, 1)$



Le conjugué d'une partition d'entiers

Une partition d'un entier n est une façon d'écrire n comme une somme d'une séquence d'entiers positifs non croissants.

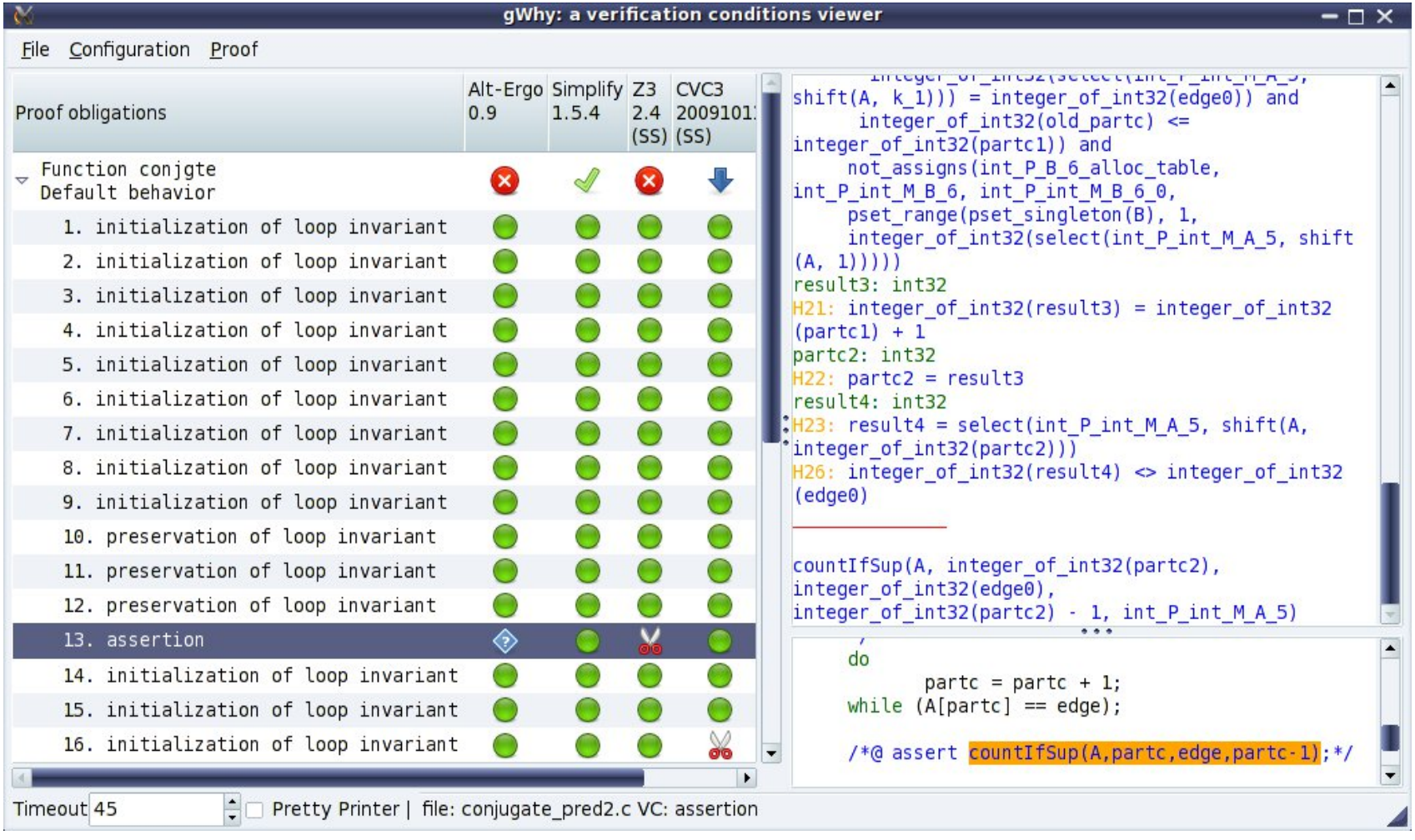
$\lambda = (3, 2, 1, 1, 1)$



Le conjugué de $(3, 2, 1, 1, 1)$ est $(5, 2, 1)$.

- Outils utilisés pour la preuve de programme :
  - Frama-C/Jessie (Why, ACSL, Logique de Hoare)
  - Prouveurs automatiques SMT (Simplify, Alt-Ergo, Z3, CVC3, ...)
  - Formats de sortie pour des prouveurs intéractifs (Coq, PVS, Isabelle/HOL, ...)
- Annotations dans le programme
  - algorithme non naïf
  - pré-requis antérieurs dans le code source
- Preuve des obligations de preuves générées

Interface graphique : gWhy



Utilisation de Coq ?

- Lorsque les prouveurs automatiquent échouent :

↪ Preuve "à la main"

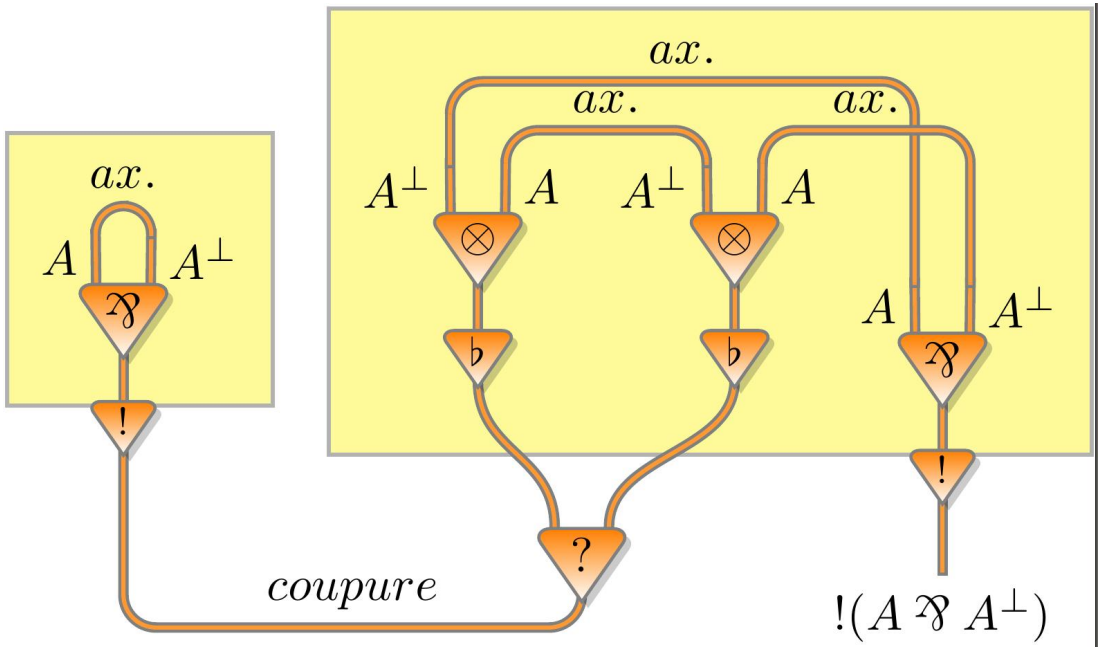
  - permet d'identifier une erreur ou un manque dans une annotation
    ↪ on corrige et on retente avec les prouveurs SMT
  - la propriété est trop compliquée pour les prouveurs SMT
    ↪ on la prouve en Coq

↪ Coq est un bon débogueur !

- Confiance dans les prouveurs SMT :
  - prouvées par au moins deux prouveurs automatiques
  - traces des prouveurs SMT ? (ex : CVC3 v3 $\rightarrow \perp$)

◂ Conclusion

Exemple d'un réseau de preuves



(Schéma P. Boudes)

retour