Utilisation de grammaires en Programmation Logique Inductive

Fabien Torre

fabien@lri.fr

Équipe Inférence et Apprentissage - LRI

JFA '96

PLI & Grammaires

- Programmation Logique Inductive :
  Étant donnés des exemples E^+ et des contre-exemples E^- pour le concept à apprendre, une théorie du domaine B, trouver une hypothèse H telle que

$$\begin{cases} B \cup H & \models & e^+ & \forall e^+ \in E^+ & (H \text{ est complète}) \\ B \cup H & \not\models & e^- & \forall e^- \in E^- & (H \text{ est correcte}) \end{cases}$$

- Grammaire en général : représentation d'un langage et génération des mots de ce langage
- Grammaires en PLI : représenter l'espace de recherche
- Ici : dénoter les **solutions** (mots du langage vérifiant certaines propriétés)

Propriétés & Grammaires attribuées

- **propriétés** : liées aux exemples ou biais de langage
- **grammaire attribuée** : grammaire + attributs + règles sémantiques
- **attributs** : description des littéraux déjà présents (fortement liée aux propriétés à satisfaire)
- **règles sémantiques** : vérification des propriétés

- Long_Max$_{\mathsf{n}}$ est vérifiée pour les clauses dont le nombre de littéraux dans le corps (taille des clauses) vaut au plus n.
- Long_Min$_{\mathsf{n}}$, la taille vaut au moins n.
- Range_Restricted, les variables de la tête de la définition doivent apparaître dans le corps.
- Vars$_{\mathsf{n}}$, n variables existentielles au maximum dans la définition.
- Prof$_{\mathsf{p}}$, on limite la profondeur des termes de la définition à p.
- Liée, les variables de la définition doivent être liées.
- Degré$_{\mathsf{d}}$, le degré des variables de la clause n'excèdent pas d.

- Les propriétés fonctionnelles nécessitent, pour chaque prédicat utilisable, sa signature en termes d'entrées et sorties. On peut alors poser des conditions sur l'utilisation, l'intanciation et l'ordre d'apparition de ces entrées et sorties.
- Bien_Typée. Les schémas typent les variables comme les prédicats. Ne sont autorisées ensuite que les clauses bien typées.
- Couvre$_{\mathsf{n}}$, la définition couvre au moins n exemples positifs.
- Écarte$_{\mathsf{n}}$, la définition écarte au moins n exemples négatifs.
- Déterminée, si tous les littéraux de la définition sont déterminés.

Construction d'une grammaire attribuée

Pour vérifier les propriétés, nous associons un attribut hérité C_h et un synthétisé C_s à chaque symbole (terminal ou non-terminal) de la grammaire hors-contexte.

Pour chaque règle $S_0 \to S_1 \dots S_n$ de la grammaire hors-contexte, on retrouve la règle sémantique suivante :

$$\begin{cases} S_0.C_s = S_n.C_s & \\ S_1.C_h = S_0.C_h & \\ S_i.C_h = S_{i-1}.C_s & \text{pour } i > 1 \\ S_i.C_s = valide(S_i, S_i.C_h) & \text{pour } S_i \text{ terminal} \end{cases}$$

Dans le cas d'une dérivation vide $S_0 \to$, on a simplement $S_0.C_s = S_0.C_h$.

Exemple de transformation

La règle

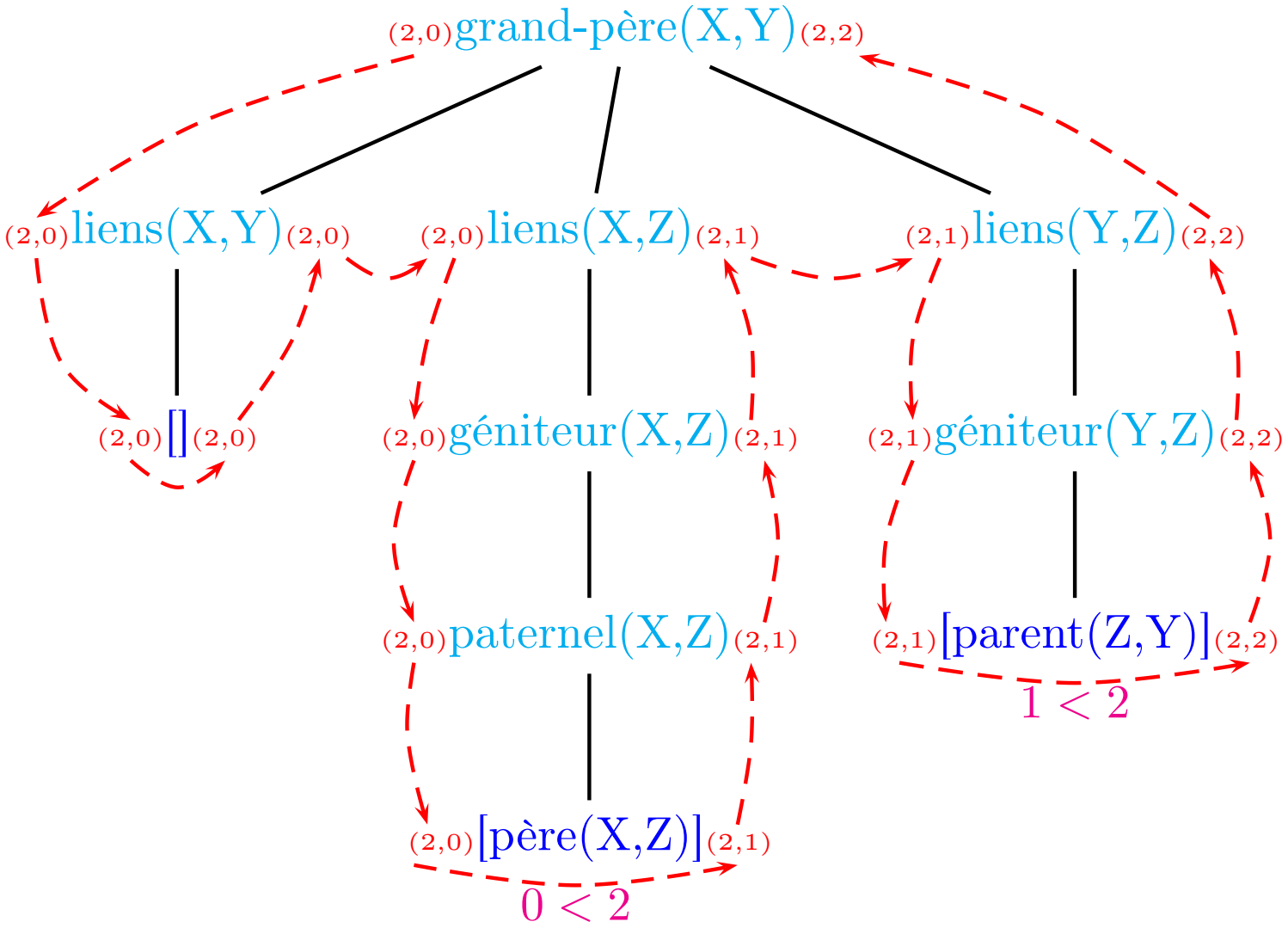
$$\text{grand-père(X,Y)} \rightarrow [\text{père(X,Z)}],\ \text{autres(X,Y,Z)}$$

sera traduite par

$$\begin{aligned} \text{grand-père(X,Y,}H\text{,}S\text{)} \rightarrow \quad & [\text{père(X,Z)}], \\ & \{\text{valide(père(X,Z),}H\text{,}I\text{)}\}, \\ & \text{autres(X,Y,Z,}I\text{,}S\text{)} \end{aligned}$$

où H,S et I sont des contextes.

Dérivation du concept grand-père



Propriétés Utilisables

Attention, comme pour les opérateurs classiques, toutes les propriétés ne sont pas utilisables.

Une propriété P sur les hypothèses est utilisable par rapport à un opérateur de raffinement $\mathcal{O}$ si et seulement si

$$\forall H, \forall H', \left.\begin{array}{r} \overline{P(H)} \\ H' \in \mathcal{O}^*(H) \end{array}\right\} \Rightarrow \overline{P(H')}$$

Propriétés utilisables & Dérivation PROLOG

Propriétés	Utilisable
Long-Max$_n$	X
Long-Min$_n$	
Range-Restricted	
Vars$_n$	X
Prof$_p$	X
Liée	X
Liée*	
Degré$_d$	X
In-Utilisés	
Out-Instanciés	

Propriétés	Utilisable
Out-Utilisés	
Out-Uniques	X
In-Instanciés	X
Bien-Typée	X
Couvre$_n$	X
Complète	X
Écarte$_n$	
Correcte	
Déterministe	X
One-Neg	X

Grammaire naïve pour grand-père

grand-père(X,Y) → littéraux([X,Y])

littéraux(L) → [].

littéraux(L) → {variables(A,B,L,NL)},
[parent(A,B)],littéraux(NL).

littéraux(L) → {variables(A,B,L,NL)},
[père(A,B)],littéraux(NL).

littéraux(L) → {variables(A,B,L,NL)},
[mère(A,B)],littéraux(NL).

Grammaire non naïve pour grand-père

grand-père(X,Y) → liens(X,Y), liens(X,Z), liens(Y,Z).

liens(A,B) → [].

liens(A,B) → géniteur(A,b).

géniteur(A,B) → [parent(A,B)].

géniteur(A,B) → [parent(B,A)].

géniteur(A,B) → paternel(A,B).

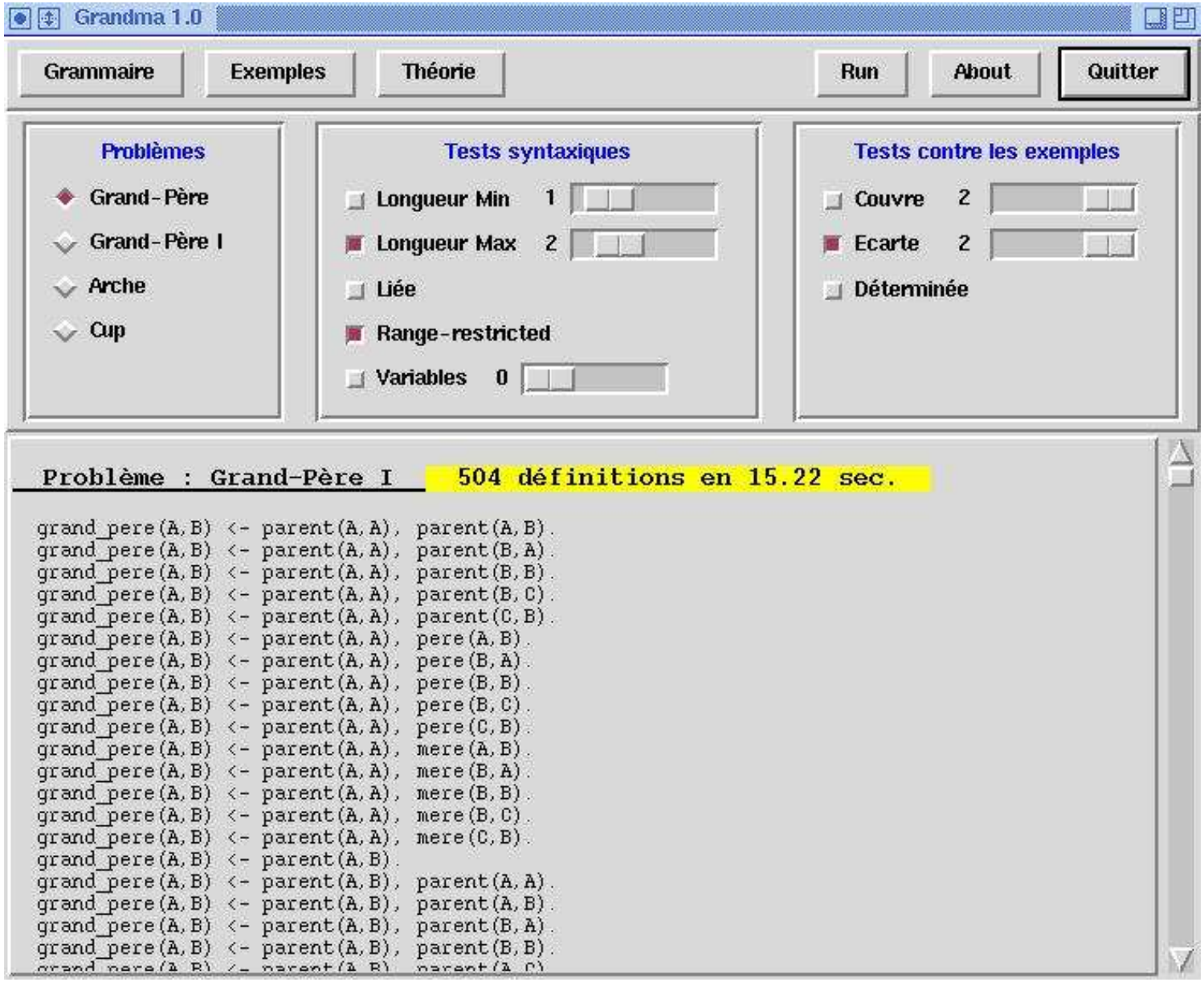
géniteur(A,B) → maternel(A,B).

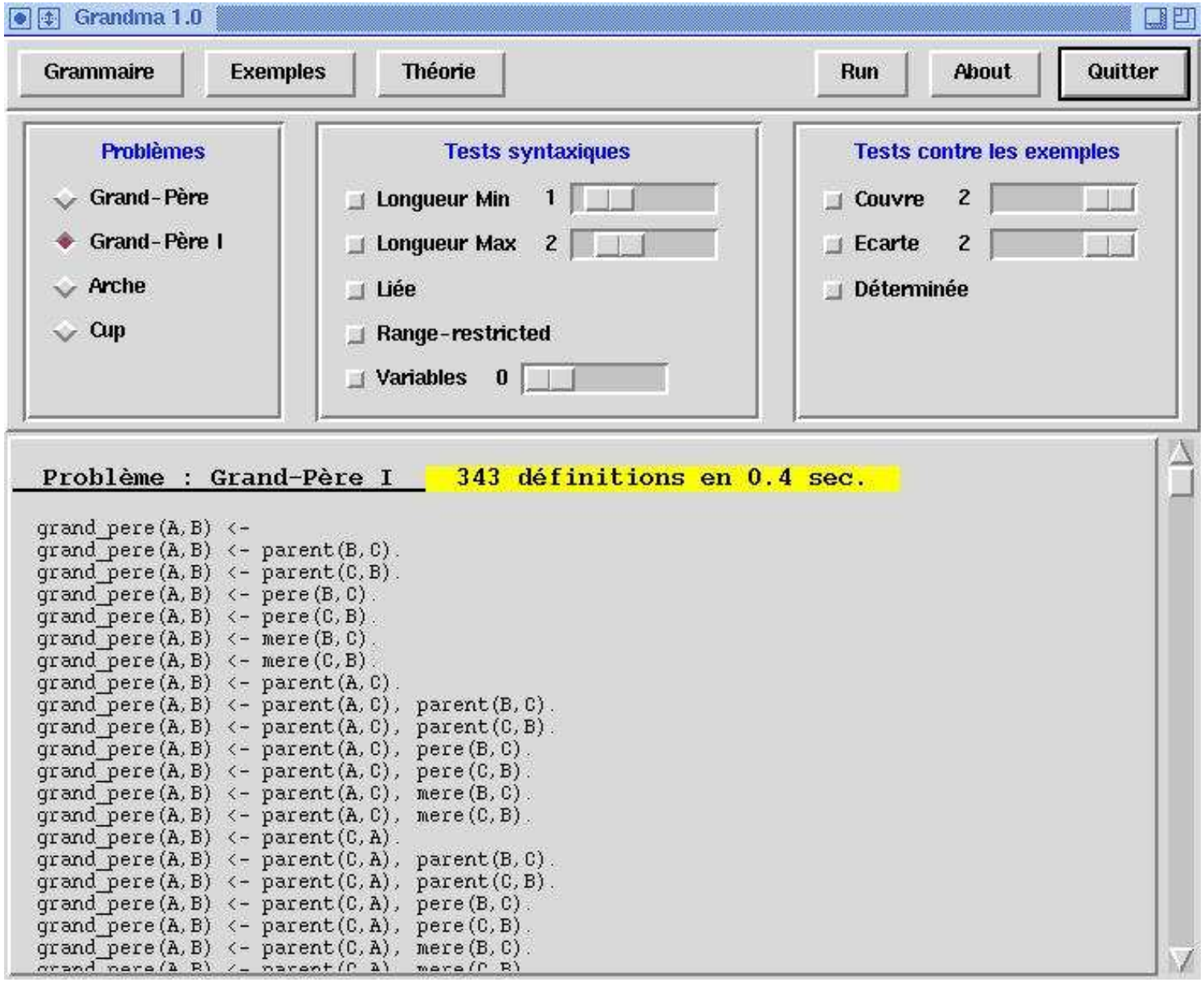
paternel(A,B) → [père(A,B)].

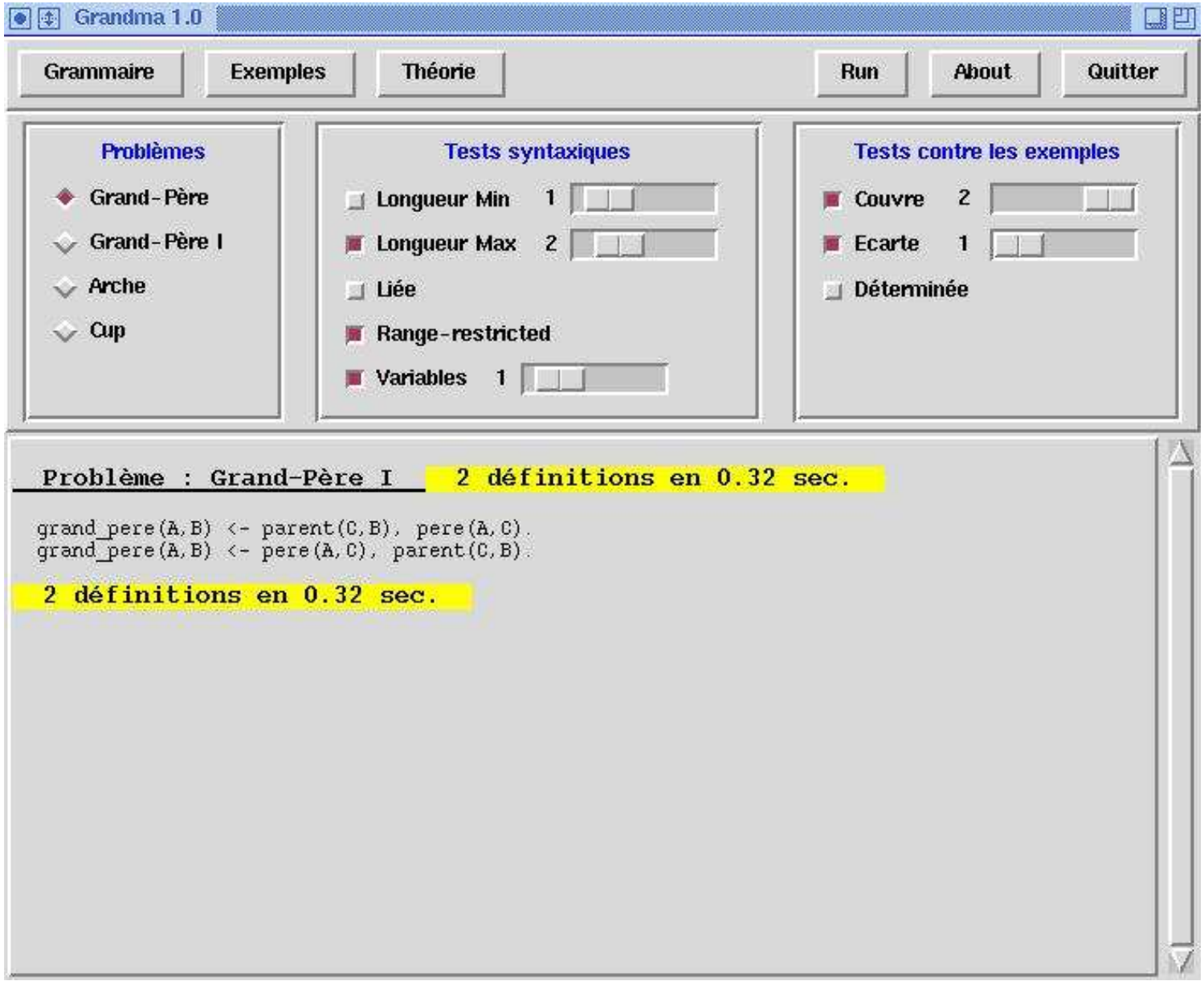
paternel(A,B) → [père(B,A)].

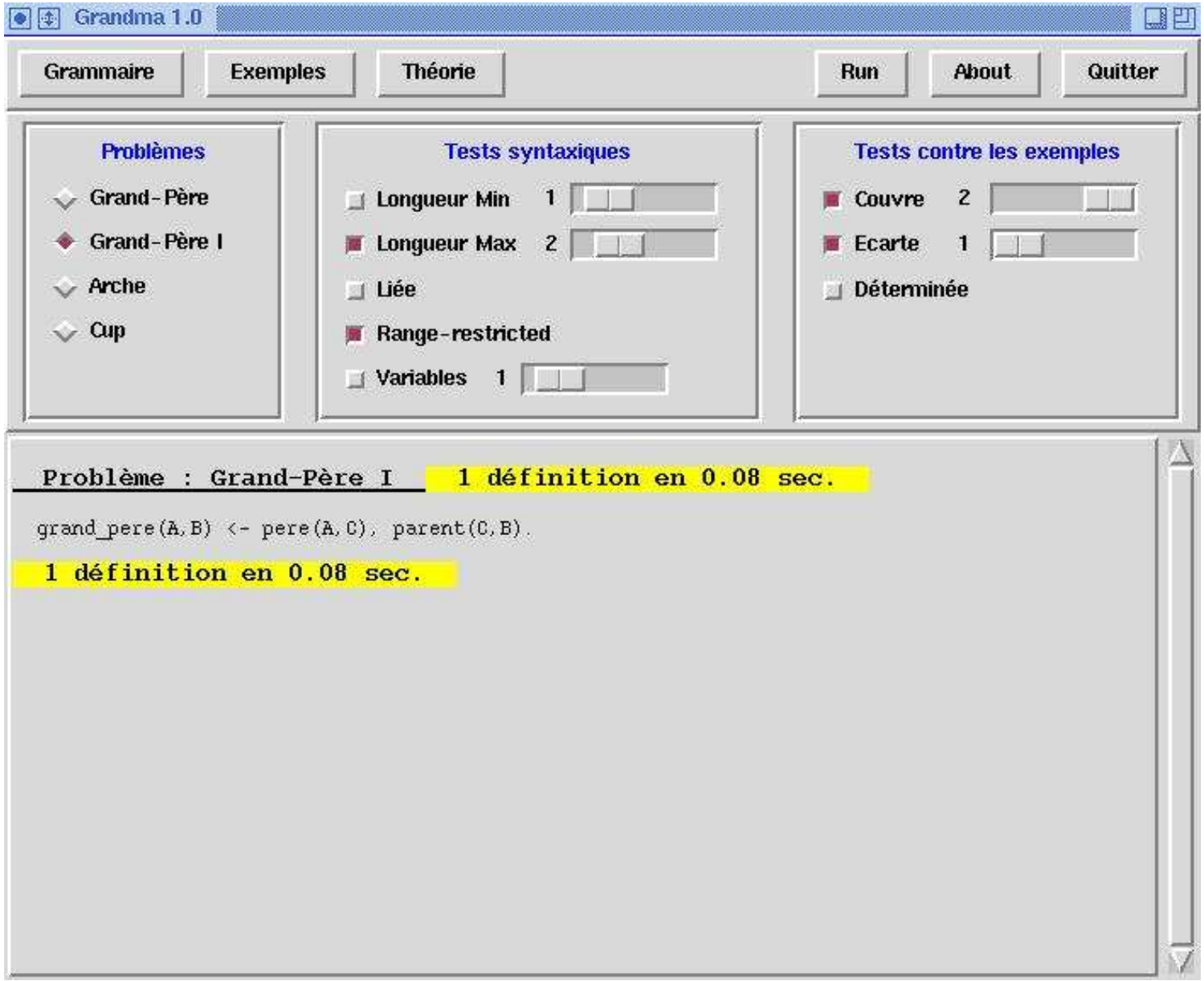
maternel(A,B) → [mère(A,B)].

maternel(A,B) → [mère(B,A)].


Grandma 1.0
Grammaire
Exemples
Théorie
Run
About
Quitter
Problèmes
Grand-Père
Grand-Père I
Arche
Cup
Tests syntaxiques
Longueur Min 1
Longueur Max 2
Liée
Range-restricted
Variables 0
Tests contre les exemples
Couvre 2
Ecarte 2
Déterminée
Problème : Grand-Père I
504 définitions en 15.22 sec.
grand_pere(A,B) <- parent(A,A), parent(A,B).
grand_pere(A,B) <- parent(A,A), parent(B,A).
grand_pere(A,B) <- parent(A,A), parent(B,B).
grand_pere(A,B) <- parent(A,A), parent(B,C).
grand_pere(A,B) <- parent(A,A), parent(C,B).
grand_pere(A,B) <- parent(A,A), pere(A,B).
grand_pere(A,B) <- parent(A,A), pere(B,A).
grand_pere(A,B) <- parent(A,A), pere(B,B).
grand_pere(A,B) <- parent(A,A), pere(B,C).
grand_pere(A,B) <- parent(A,A), pere(C,B).
grand_pere(A,B) <- parent(A,A), mere(A,B).
grand_pere(A,B) <- parent(A,A), mere(B,A).
grand_pere(A,B) <- parent(A,A), mere(B,B).
grand_pere(A,B) <- parent(A,A), mere(B,C).
grand_pere(A,B) <- parent(A,A), mere(C,B).
grand_pere(A,B) <- parent(A,B).
grand_pere(A,B) <- parent(A,B), parent(A,A).
grand_pere(A,B) <- parent(A,B), parent(A,B).
grand_pere(A,B) <- parent(A,B), parent(B,A).
grand_pere(A,B) <- parent(A,B), parent(B,B).


Grandma 1.0
Grammaire
Exemples
Théorie
Run
About
Quitter
Problèmes
Grand-Père
Grand-Père I
Arche
Cup
Tests syntaxiques
Longueur Min 1
Longueur Max 2
Liée
Range-restricted
Variables 0
Tests contre les exemples
Couvre 2
Ecarte 2
Déterminée
Problème : Grand-Père I
343 définitions en 0.4 sec.
grand_pere(A,B) <-
grand_pere(A,B) <- parent(B,C).
grand_pere(A,B) <- parent(C,B).
grand_pere(A,B) <- pere(B,C).
grand_pere(A,B) <- pere(C,B).
grand_pere(A,B) <- mere(B,C).
grand_pere(A,B) <- mere(C,B).
grand_pere(A,B) <- parent(A,C).
grand_pere(A,B) <- parent(A,C), parent(B,C).
grand_pere(A,B) <- parent(A,C), parent(C,B).
grand_pere(A,B) <- parent(A,C), pere(B,C).
grand_pere(A,B) <- parent(A,C), pere(C,B).
grand_pere(A,B) <- parent(A,C), mere(B,C).
grand_pere(A,B) <- parent(A,C), mere(C,B).
grand_pere(A,B) <- parent(C,A).
grand_pere(A,B) <- parent(C,A), parent(B,C).
grand_pere(A,B) <- parent(C,A), parent(C,B).
grand_pere(A,B) <- parent(C,A), pere(B,C).
grand_pere(A,B) <- parent(C,A), pere(C,B).
grand_pere(A,B) <- parent(C,A), mere(B,C).


Grandma 1.0
Grammaire
Exemples
Théorie
Run
About
Quitter
Problèmes
Grand-Père
Grand-Père I
Arche
Cup
Tests syntaxiques
Longueur Min 1
Longueur Max 2
Liée
Range-restricted
Variables 1
Tests contre les exemples
Couvre 2
Ecarte 1
Déterminée
Problème : Grand-Père I
2 définitions en 0.32 sec.
grand_pere(A,B) <- parent(C,B), pere(A,C).
grand_pere(A,B) <- pere(A,C), parent(C,B).
2 définitions en 0.32 sec.


Grandma 1.0
Grammaire
Exemples
Théorie
Run
About
Quitter
Problèmes
Grand-Père
Grand-Père I
Arche
Cup
Tests syntaxiques
Longueur Min 1
Longueur Max 2
Liée
Range-restricted
Variables 1
Tests contre les exemples
Couvre 2
Ecarte 1
Déterminée
Problème : Grand-Père I
1 définition en 0.08 sec.
grand_pere(A,B) <- pere(A,C), parent(C,B).
1 définition en 0.08 sec.