Compte-rendu du GdT "Intégration"

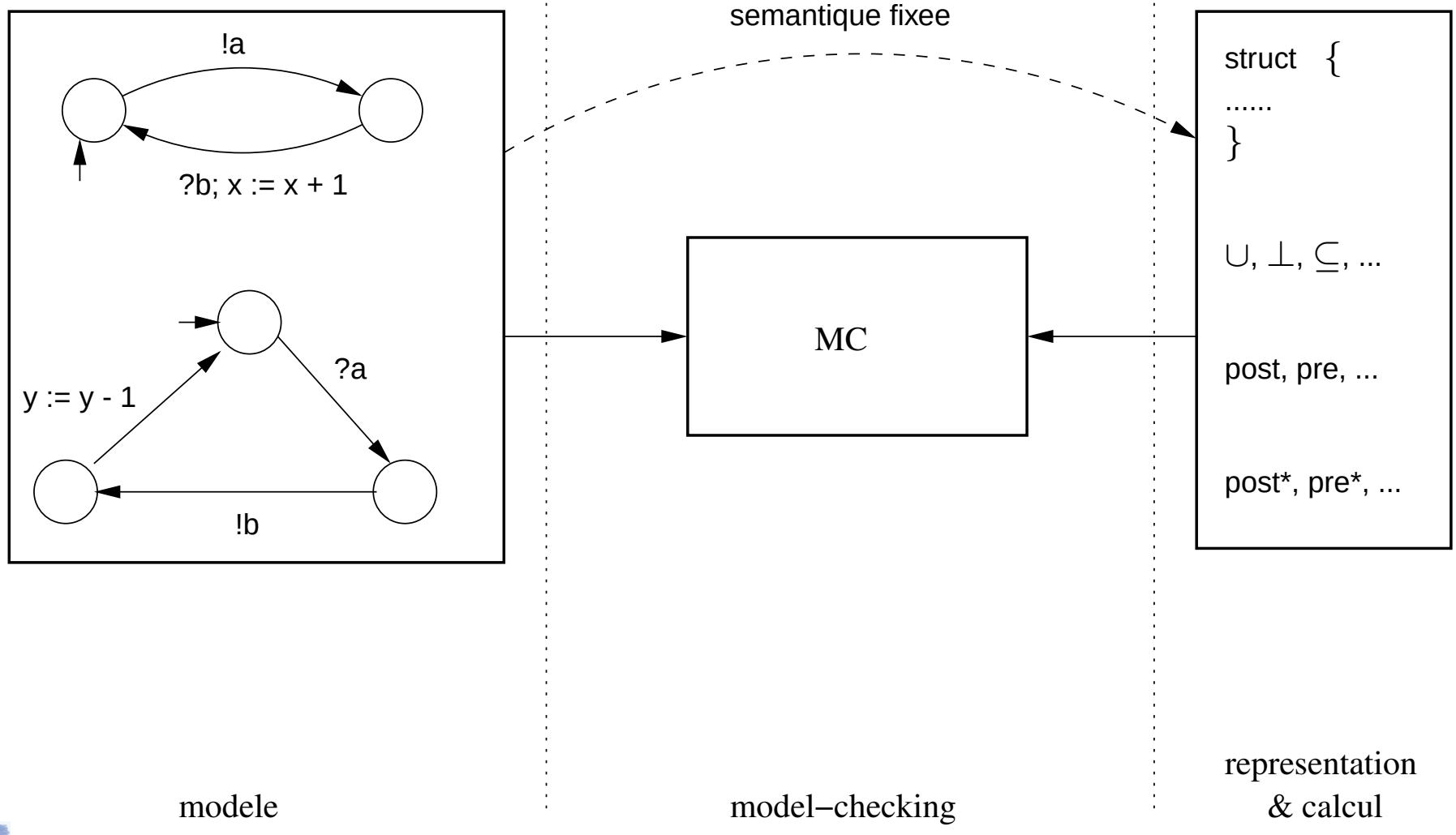
F. Herbreteau, S. Bardin, A. Finkel, D. Nowak, M. Sighireanu, G. Sutre
et tous ceux qui ont contribué à FAST, TREX et VERIFAST

Journées PERSÉE - 14/15 septembre 2004
LaBRI, Bordeaux

Objectifs

- Intégration de model-checkers accélérés :
  - FAST : accélération d'automates à compteurs;
  - TREX : interpolation d'automates étendus : file + compteurs + temps.
- Mettre à profit les réflexions du projet VERIFAST :
  - plateforme d'expérimentation générique et modulaire.
- On veut donc un model-checker :
  - générique : pas de modèle fixé;
  - et modulaire : combiner les techniques (FAST et TREX par exemple).

Model-checker "traditionnel"



Modèle générique

- Système à contraintes ALTARICA : $(\overrightarrow{s}, \overrightarrow{f}, E, A, \mathtt{Op}, T)$
  - $\overrightarrow{s}, \overrightarrow{f}$: variables d'états/de flux, E : événements, $A(\overrightarrow{s}, \overrightarrow{f})$: assertion, $\mathtt{Op}$: opérations
  - T : transitions $G(\overrightarrow{s}, \overrightarrow{f}) \xrightarrow{e} \overrightarrow{s} := \rho(\overrightarrow{s}, \overrightarrow{f}, \mathtt{Op})$
- Sémantique des données : $\delta : \mathbb{D}^{\overrightarrow{s}, \overrightarrow{f}} \times \mathtt{Op} \rightarrow \mathbb{D}^{\overrightarrow{s}}$
- Sémantique opérationnelle : $(S \times F, \mathtt{Op}, \rightarrow)$
  - $S = \{s \in \mathbb{D}^{\overrightarrow{s}} \mid (\exists f \in \mathbb{D}^{\overrightarrow{f}})\ (s, f) \in A\}$ et $F = \mathbb{D}^{\overrightarrow{f}}$,
  - $\rightarrow \subseteq (S \times F) \times \mathtt{Op} \times (S \times F)$ définie par :

$$(s, f) \xrightarrow{\mathtt{op}} (s', f') \text{ si } (s, f) \in A \text{ et } s' = \delta(s, f, \mathtt{op}) \text{ et } f' \in A$$

- Généricité : domaine $\mathbb{D}$ et opérations $\mathtt{Op}$.

Le langage de spécification

- **Contrôle** : spécification ALTARICA
  - modélisation de processus avec synchronisation "à la Arnold&Nivat";
  - bonus : hiérarchie, priorités entre événements;
  - outils (mise à plat, MEC V, altatools, . . .);
  - développé dans un laboratoire membre du projet, et de façon ouverte;
  - utilisation industrielle.

- **Généricité** des types : signatures

```
sig int :
  0, 1 : int
  + : int * int -> int
  <= : int * int -> bool
  ...
end
```

Format des transitions

- Débat entre :
  - **affectations gardées** :

$$G(\overrightarrow{s}, \overrightarrow{f}) \rightarrow \overrightarrow{s} := \rho(\overrightarrow{s}, \overrightarrow{f}, \mathtt{Op})$$

    - proche de la spécification.

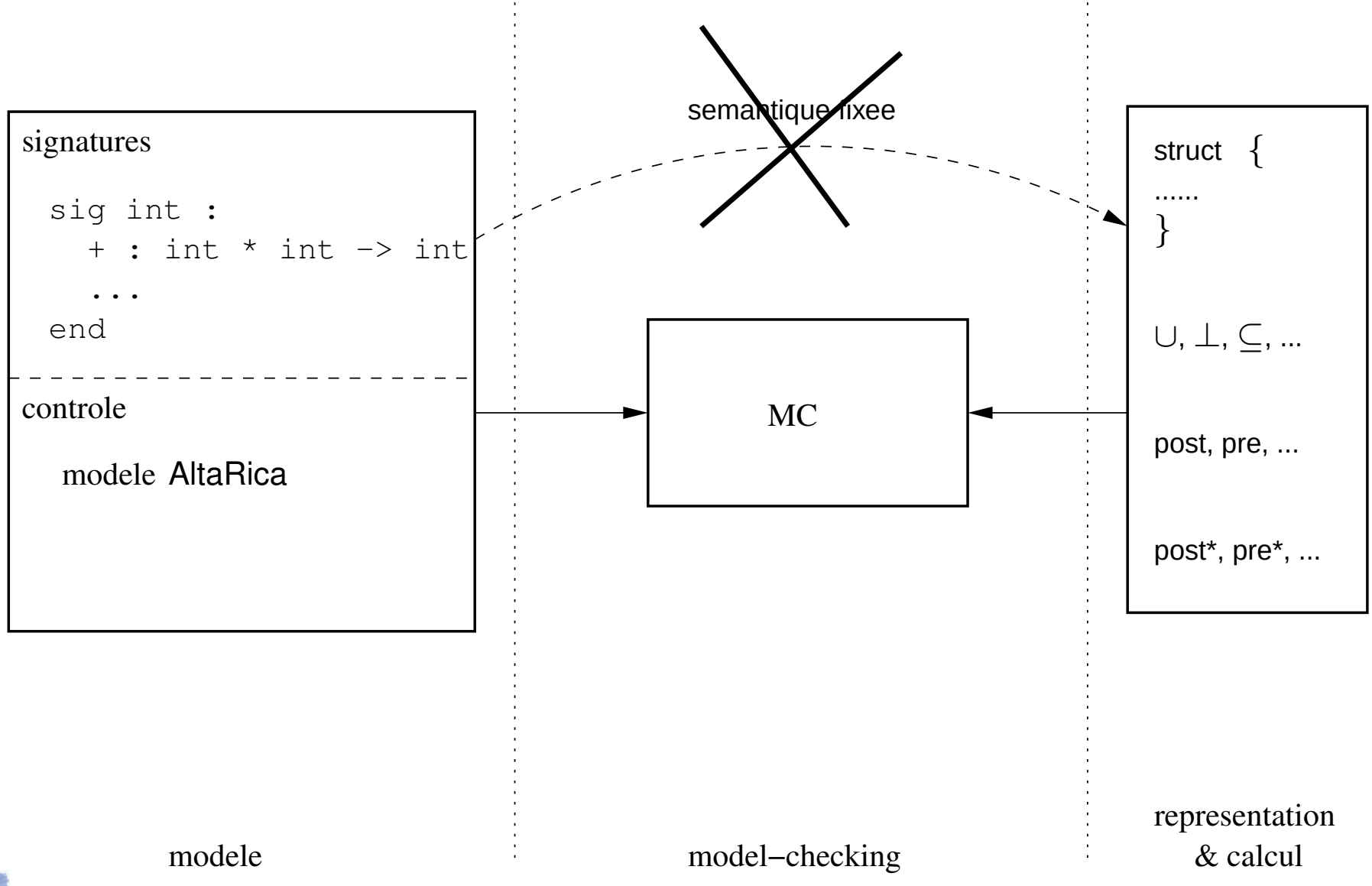
  - **relations binaires** :

$$\phi(\overrightarrow{s}, \overrightarrow{f}, \overrightarrow{s}')$$

    - modélise les affectations gardées;
    - ajout de méta-transitions dans le modèle;
    - modélisation du temps quantitatif.

- Les transitions d'AltaRica sont des affectations gardées.

Architecture générique : 1ère étape



Représentation et calcul

- **Calcul symbolique** d'ensembles d'états : régions et opérations sur celles-ci.

- **Régions** : représentation d'ensemble d'états et opérations ensemblistes (ex : QDD, DBM, ...).

- **Représentations symboliques** : calcul de `post`/`pre` sur des régions.
  - sémantique des opérations `Op` !
  - problèmes liés à l'évaluation des opérations (`Op`) !

- **Accélérations, interpolations** : calcul de `post*` et `pre*` sur des régions.

Représentation et calcul

- **Calcul symbolique** d'ensembles d'états : régions et opérations sur celles-ci.

- **Régions** : représentation d'ensemble d'états et opérations ensemblistes (ex : QDD, DBM, ...).

- **Représentations symboliques** : calcul de `post`/`pre` sur des régions.
  - sémantique des opérations `Op` !
  - problèmes liés à l'évaluation des opérations (`Op`) !

- **Accélérations, interpolations** : calcul de `post*` et `pre*` sur des régions.

Séparation voulue pour la modularité

Les régions

- **Objectifs** : structure de données et opérations ensemblistes sur celle-ci pour les algorithmes de model-checking.

$$(V, R, \bot, \sqcup, \sqsubseteq, [\![\cdot]\!])$$

  - V ensemble de variables représentées;
  - R est un ensemble de régions, interprétée par $[\![\cdot]\!] : R \to 2^{\mathbb{D}^V}$;
  - $\bot$ est la région vide ($[\![\bot]\!] = \emptyset$);
  - $\sqcup$ est l'union de régions ($[\![r \sqcup r']\!] = [\![r]\!] \cup [\![r']\!]$);
  - $\sqsubseteq$ est le pré-ordre d'inclusion induit : $r \sqsubseteq r'$ ssi $[\![r]\!] \subseteq [\![r']\!]$.

- On peut ajouter $\top$, $\sqcap$, etc.

- Au niveau de l'**implémentation**, il est nécessaire de pouvoir transformer une chaîne de caractères en région (ex : "$x \leq 2$" en NDD).

Les représentations symboliques

- **Objectifs** : évaluation des opérations ($\texttt{Op}$) sur les régions.
- Sémantique symbolique des données : $\hat{\delta} : R \times \texttt{Op} \to R$ telle que

$$[\![\hat{\delta}(r, \texttt{op})]\!] = \{(s', f') \in \mathbb{D}^{\overrightarrow{s}, \overrightarrow{s}} \mid (\exists (s, f) \in [\![r]\!])\ s' = \delta((s, f), \texttt{op}) \wedge f' \in A\}$$

- Pour une transition $G(\overrightarrow{s}, \overrightarrow{f}) \xrightarrow{e} \overrightarrow{s} := \rho(\overrightarrow{s}, \overrightarrow{f}, \texttt{Op})$:

$$\texttt{post}(r, e) = \hat{\delta}(r \sqcap r_G, \rho)$$

où r_G est la région correspondant à G.

  - **gestion des erreurs** (ex : perte de précision, division par 0) par exception + retour d'une surapproximation de $\texttt{post}(r, e)$.

- On peut définir $\texttt{pre}$ de la même manière.

Les accélérations (1/2)

- **Généricité** : traiter les accélérations, les interpolations, le widening, etc.

- **Accélération exacte** d'une séquence σ de transitions : fonction $\hat{\sigma}_A : R \to R$ telle que

$$[\![\hat{\sigma}_A(r,\sigma)]\!] = \{(s',f') \in \mathbb{D}^{\overrightarrow{s},\overrightarrow{f}} \mid (\exists (s,f) \in [\![r]\!])(\exists i \in \mathbb{N})\ s' = \delta(\sigma^i,(s,f)) \wedge f' \in A$$

- **Interpolation** d'une séquence σ telle que $r' = \delta(\sigma, r)$ et $r'' = \delta(\sigma, r')$, et $r' - r = r'' - r' = \Delta$: fonction $\hat{\sigma}_I : R \to R$ telle que

$$[\![\hat{\sigma}_I(r,r',r'',\sigma)]\!] = \{(s',f') \in \mathbb{D}^{\overrightarrow{s},\overrightarrow{f}} \mid (\exists (s,f) \in [\![r]\!])(\exists i \in \mathbb{N})\ s' = s + i.\Delta \wedge \Delta = r' - r \wedge f' \in A\}$$

$$\supseteq [\![\hat{\sigma}_A(r,\sigma)]\!]$$

Les accélérations (2/2)

- **Widening** pour une séquence σ telle que $r \xrightarrow{\sigma} r'$: fonction $\hat{\sigma}_W : R \to R$ telle que

$$[\![\hat{\sigma}_W(r, r', \sigma)]\!] = \{(s', f') \in \mathbb{D}^{\overrightarrow{s}, \overrightarrow{f}} \mid (s', f') \in r \nabla r'\}$$
$$\supseteq [\![\hat{\sigma}_A(r, \sigma)]\!]$$

- On considère d'une façon générale qu'une **fonction d'accélération** pour une séquence σ est une fonction $\hat{\sigma} : (R_i)_{0 \leq i \leq n} \times R \to R$ telle que

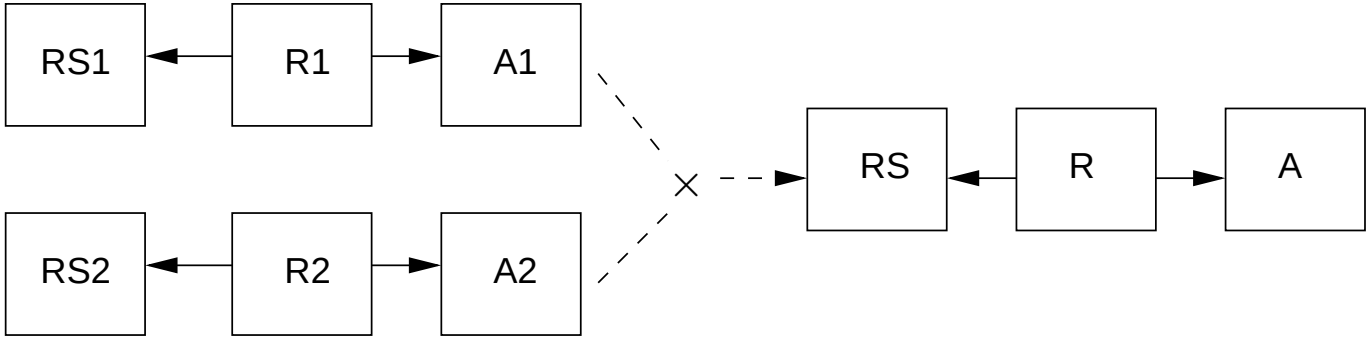
$$[\![\hat{\sigma}((r_i)_{0 \leq i \leq n}, \sigma)]\!] \supseteq [\![\hat{\sigma}_A(r_0, \sigma)]\!]$$

où $(R_i)_{0 \leq i \leq n}$ désigne l'ensemble des suites finies de régions telles que $R_i = \delta(\sigma, R_{i-1})$ pour tout $i \geq 1$.

- Extension à l'**accélération d'un langage régulier** L plutôt qu'une séquence σ.

Composition

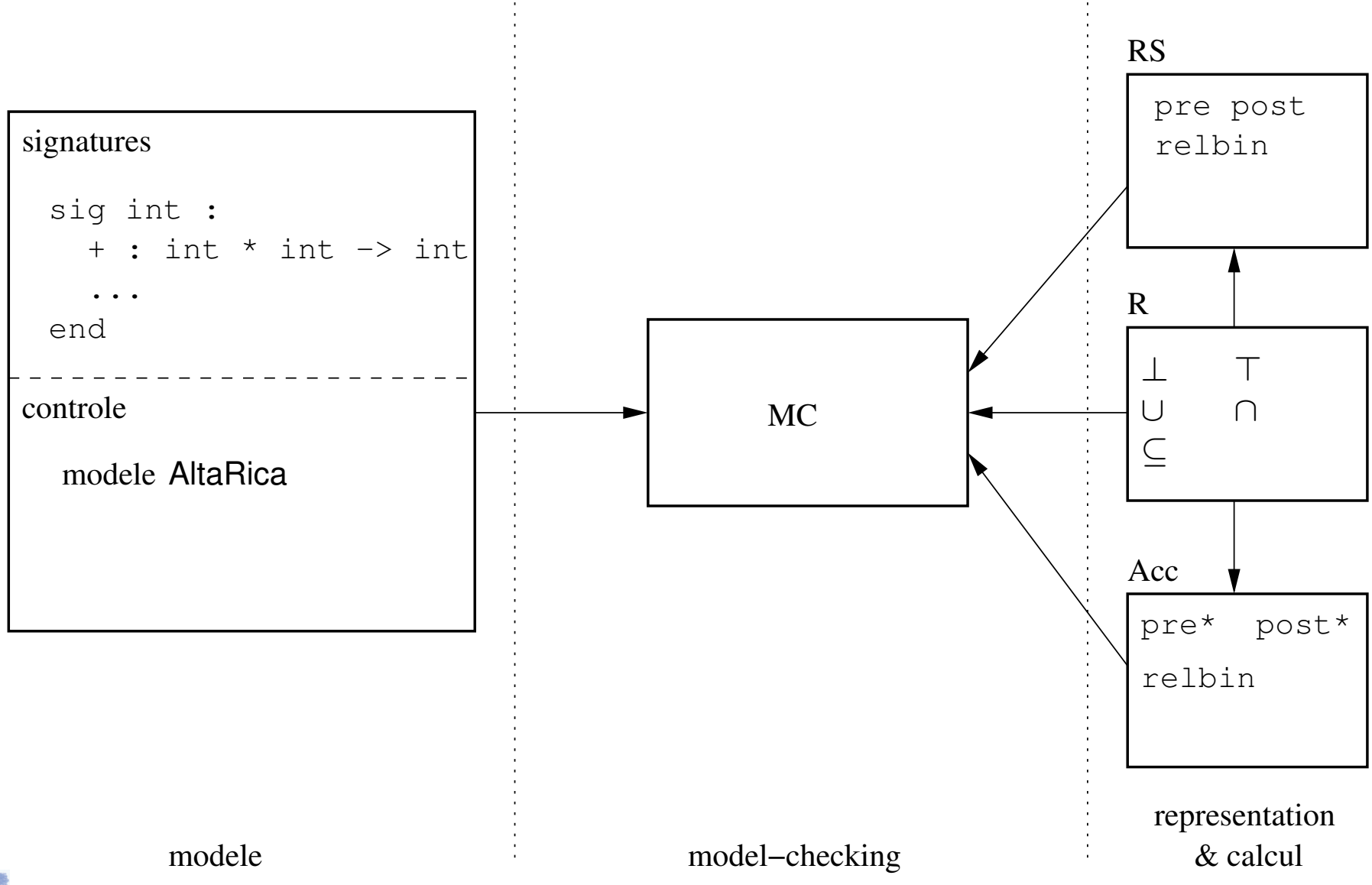
- Pour l'analyse des **systèmes hétérogènes** : il faut composer les régions, les représentations symboliques et les accélérations.



- Le **produit cartésien** (orthogonal) peut être défini génériquement.

- S'il y a de l'**information partagée** : définir un produit *ad hoc* (ex : les paramètres de TREX).

Architecture générique : 2ème étape



Fonctions du model-checker

- **Traditionnellement** :
  - lien entre le modèle (syntaxe) et les outils symboliques (sémantique).
  - fonctions de calcul de `post*`/`pre*` global.
  - heuristiques d'analyse.
  - configuration et assistance à l'utilisateur

- Pour **préserver la modularité**, on ne peut pas avoir cette approche "fourre-tout". On distingue 2 aspects :
  - les outils automatiques (ex : calcul de `post*` pour le modèle);
  - l'expertise de l'utilisateur (ex : configuration des options du model-checker).

- **Souhait** : développer un assistant de vérification.

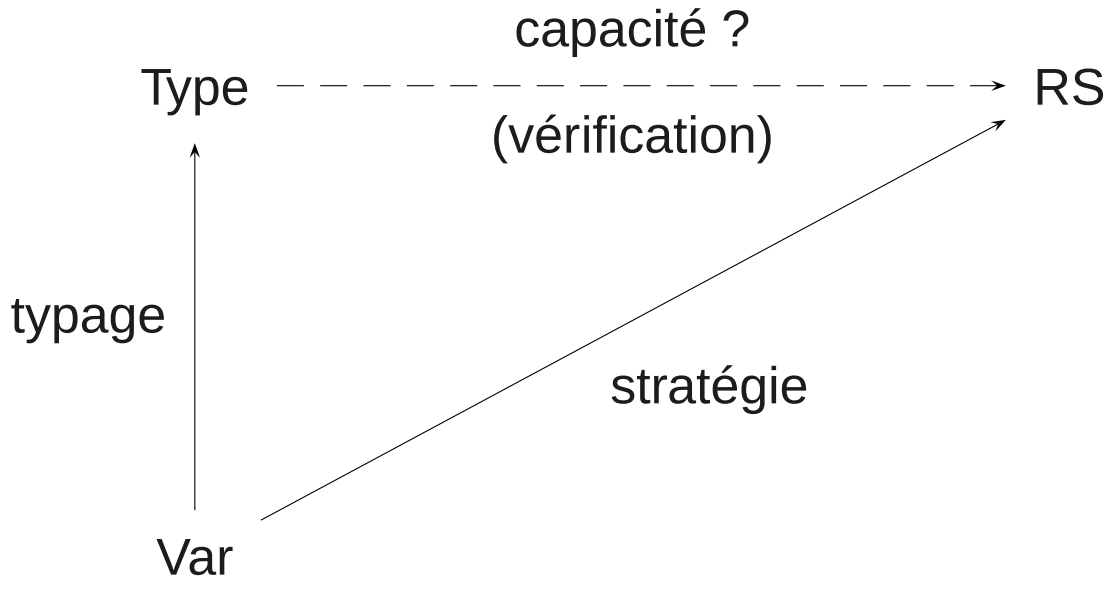
Intégrer l'expertise de l'utilisateur

- **Expérience** : l'approche "presse-bouton" n'est pas efficace.
- La vérification a besoin d'être paramétrée, guidée, orientée par l'expert.
- Deux aspects :
  - **diagnostique** : assistance à l'utilisateur (ex : proposition de représentation adaptée dans TREX);
  - **stratégie** : programmation de la vérification "à la HYTECH" (ex : quelle heuristique utiliser, ordre des calculs, etc).
- **Solution** : langage de script et boucle interactive.

```
...
> let reach_dir = forward;;
> let Rinit = region_of_string(''x<= 2 && y = 7'');;
> cap (post*(Rinit)) Rbad;;
   empty : int
>
...
```

Sémantique du modèle

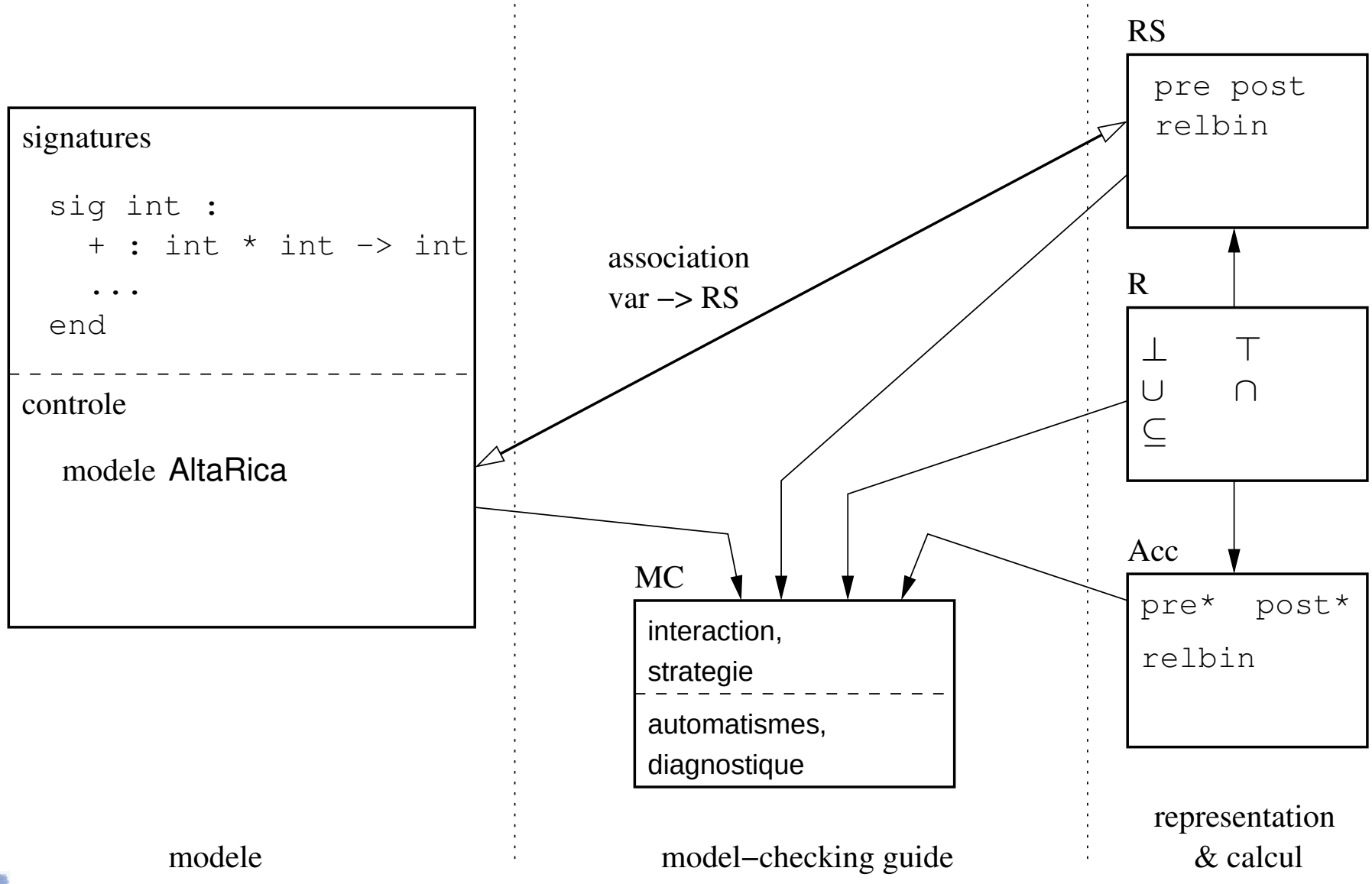
- fixée par l'utilisateur par association **"variable - représentation symbolique"** :



- permet de choisir la représentation la mieux adaptée au modèle traité.

- c'est une composante de la stratégie de vérification.

Architecture générique : proposition



Conclusions

- **Notre projet** est de développer un assistant de vérification accélérée modulaire et générique.
- **Validation** de nos choix.
- Planning (étapes) :
  - formaliser l'architecture;
  - programmer l'interface (OCAML);
  - écrire le rapport ("deliverable");
  - programmation de l'assistant;
  - intégration des outils existants.