Event-B to lambdapi

Jean-Paul Bodeveix, Anne Grieu

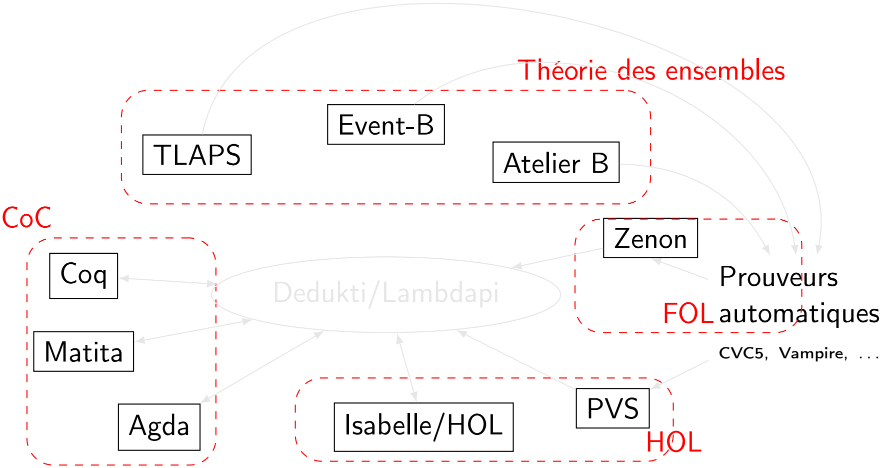
INP - IRIT
Université de Toulouse
Équipe ACADIE

JFLA 2025
Roiffé

ACADIE

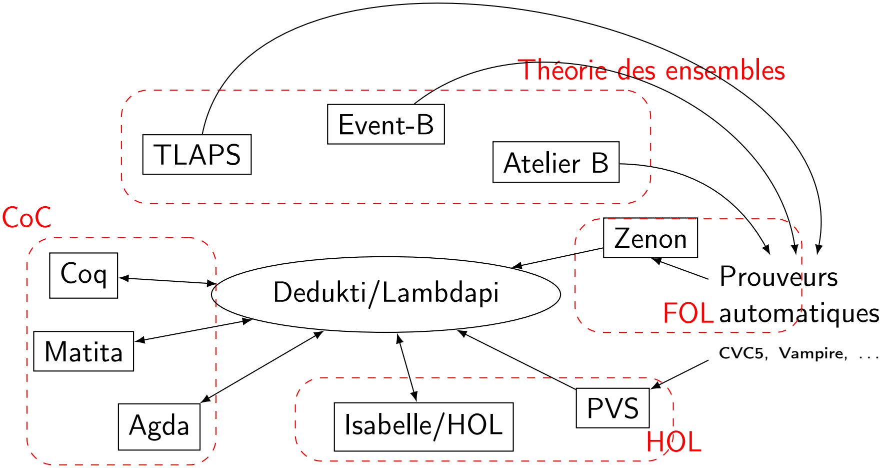
Projet ANR ICSPA

Méthodes formelles - Interopérabilité



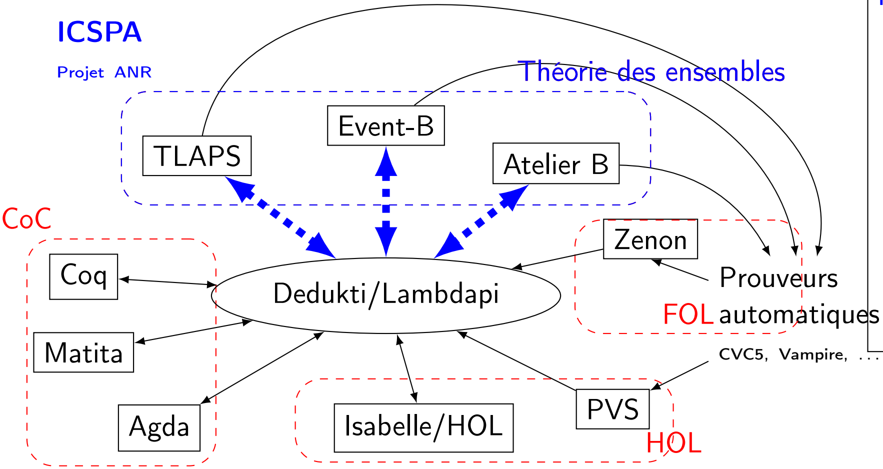
ACADIE

Méthodes formelles - Interopérabilité



1. europroofnet.github.io/_pages/WG1/Jun2022/frederic.pdf

Méthodes formelles - Interopérabilité



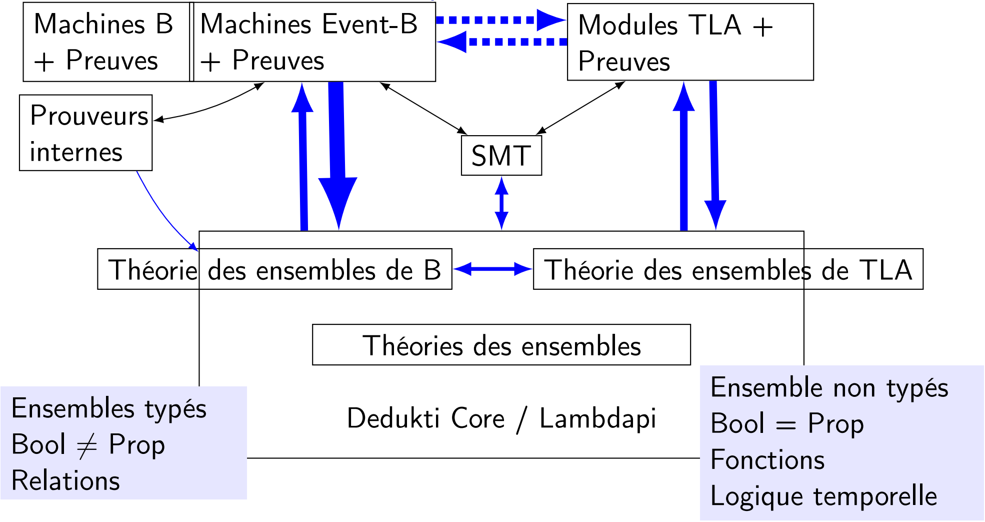
Partenaires ICSPA

- SAMOVAR
- INRIA Nancy
- INRIA Paris-Saclay
- IRIT
- LIRMM
- CLEARSY

ACADIE [1]

1. `europroofnet.github.io/_pages/WG1/Jun2022/frederic.pdf`

Méthodes formelles reposant sur la théorie des ensembles



ACADIE

Plonger Event-B dans Lambdapi

Théorème de Cantor

Il n'existe pas de fonction surjective d'un ensemble vers l'ensemble de ses parties.

Dans Rodin

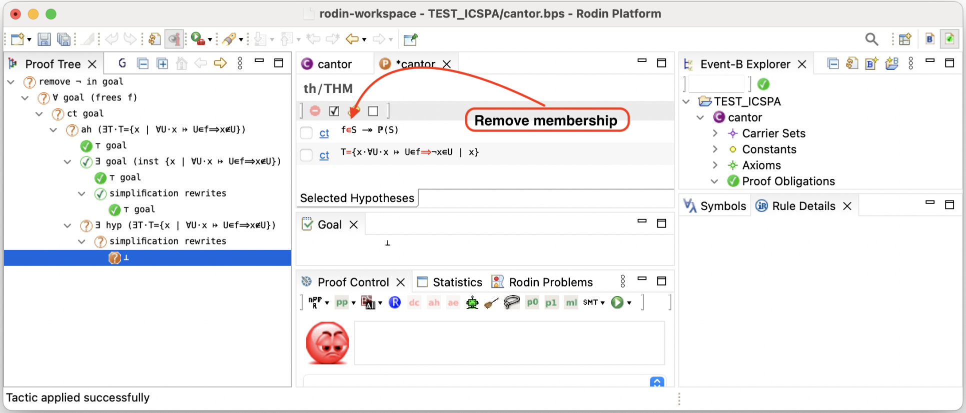
```
cantor
context cantor

sets S

axioms
  theorem @th ¬(∃f·f∈S↠ℙ(S))
end
```

Traduire ce contexte et la preuve générée par Rodin :

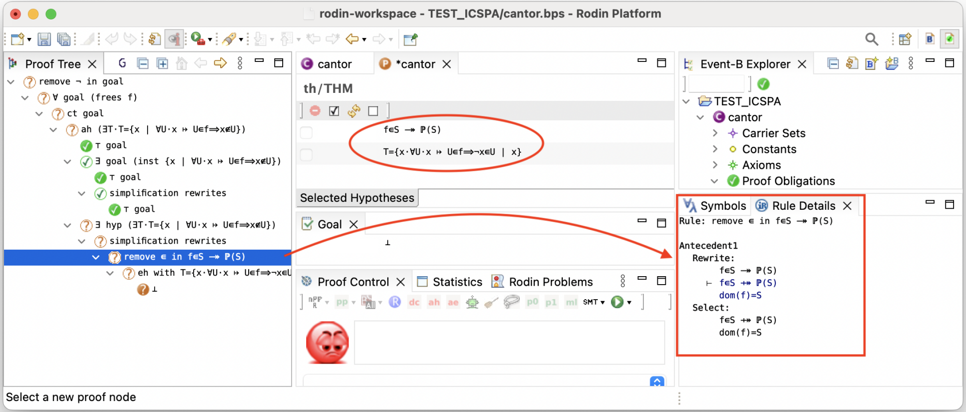
- traduction du langage mathématique
- traduction de l'arbre de preuve.

ACADIE

Prouveur interactif



Prouveur interactif



Arbre de preuve

Proof Tree

```
remove ¬ in goal
  ∀ goal (frees f)
    ct goal
      remove ∈ in f∈S ↠ ℙ(S)
        remove ∈ in f∈S ⤀ ℙ(S)
          remove ∈ in f∈S ⇸ ℙ(S)
            type rewrites
              rewrites set equality in hyp (ran(f)=ℙ(S))
                type rewrites
                  remove ⊆ in ℙ(S)⊆ran(f)
                    type rewrites
                      simplification rewrites
                        ∀ hyp (inst {x | ∀U·x ↦ U∈f⇒¬x∈U})
                          ⊤ goal
                          remove ∈ in {x | ∀U·x ↦ U∈f⇒¬x∈U}∈ran(f)
                            ∃ hyp (∃x·x ↦ {x | ∀U·x ↦ U∈f⇒¬x∈U}∈f)
                              dc (x∈{x | ∀U·x ↦ U∈f⇒¬x∈U})
                                ⊤ goal
```

ACADIE

Théorie mathématique de B/Event-B

C'est une logique classique du premier ordre, étendue par une théorie des ensembles typée, au-dessus de laquelle est définie l'arithmétique.

Méthode d'Abrial :

- Constructeurs de base avec une axiomatique permettant une procédure de décision
- Constructeurs dérivés, définis à partir des constructeurs de base
- Règles dérivées, prouvées à partir des précédentes

Calcul des propositions

Constructeurs de base

1. $\wedge, \Rightarrow, \neg$

   $\rightarrow$ Théorie axiomatique

2. Constante $\perp$ + une expression plus pratique des règles définissant les constructeurs.

   $\rightarrow$Théorie déterministe

	Antécédents	Conséquent
R1	$H \vdash P$ $H \vdash Q$	$H \vdash P \wedge Q$
R2	$H \vdash P \wedge Q$	$H \vdash P$
R3	$H \vdash P \wedge Q$	$H \vdash Q$
R4	$H, P \vdash Q$	$H \vdash P \Rightarrow Q$
R5	$H \vdash P \Rightarrow Q$	$H, P \vdash Q$
R6	$H, \neg Q \vdash P$ $H, \neg Q \vdash \neg P$	$H \vdash Q$
R7	$H, Q \vdash P$ $H, Q \vdash \neg P$	$H \vdash \neg Q$

ACADIE

Calcul des propositions

Constructeurs de base

1. $\wedge, \Rightarrow, \neg$
   $\rightarrow$ Théorie axiomatique
2. Constante $\perp$ + une expression plus pratique des règles définissant les constructeurs.
   $\rightarrow$Théorie déterministe

	Antécédents	Conséquent
INI	$H \vdash \neg R \Rightarrow \perp$	$H \vdash R$
AXM		$H, P, \neg P \vdash R$
AND1	$H \vdash \neg Q \Rightarrow R$ $H \vdash \neg P \Rightarrow R$	$H \vdash \neg(P \wedge Q) \Rightarrow R$
AND2	$H \vdash P \Rightarrow (Q \Rightarrow R)$	$H \vdash (P \wedge Q) \Rightarrow R$
IMP1	$H \vdash P \Rightarrow (\neg Q \Rightarrow R)$	$H \vdash \neg(P \Rightarrow Q) \Rightarrow R$
IMP2	$H \vdash Q \Rightarrow R$ $H \vdash \neg P \Rightarrow R$	$H \vdash (P \Rightarrow Q) \Rightarrow R$
NEG	$H \vdash P \Rightarrow R$	$H \vdash \neg\neg P \Rightarrow R$
DED	$H, P \vdash R$	$H \vdash P \Rightarrow R$

Ordre des règles : AXM, IMP1, IMP2, AND1, AND2, NEG
Procédure de preuve : INI ; (RULES* ;DED)*

ACADIE

Calcul des propositions

Constructeurs dérivés

$\vee$, $\Leftrightarrow$ et $\top$ sont définis comme des réécritures des constructeurs de base.

Prédicat	Définition
$\top$	$\neg\bot$
$P \vee Q$	$\neg P \Rightarrow Q$
$P \Leftrightarrow Q$	$(P \Rightarrow Q) \wedge (Q \Rightarrow P)$

Règles dérivées

Prouvées à partir des règles précédentes.

...

	Antécédents	Conséquent
OR1	$H \vdash \neg P \Rightarrow (\neg Q \Rightarrow R)$	$H \vdash \neg(P \vee Q) \Rightarrow R$
OR2	$H \vdash Q \Rightarrow R$ $H \vdash P \Rightarrow R$	$H \vdash (P \vee Q) \Rightarrow R$
EQV1	$H \vdash P \Rightarrow (\neg Q \Rightarrow R)$ $H \vdash \neg P \Rightarrow (Q \Rightarrow R)$	$H \vdash (\neg P \Leftrightarrow Q) \Rightarrow R$
EQV2	$H \vdash P \Rightarrow (Q \Rightarrow R)$ $H \vdash \neg P \Rightarrow (\neg Q \Rightarrow R)$	$H \vdash (P \Leftrightarrow Q) \Rightarrow R$

Lambdapi

*« Lambdapi is an interactive proof system featuring **dependent types** like in Martin-Lőf's type theory, but allowing to define objects and types using **oriented equations**, aka **rewriting rules**, and reason modulo those equations. »* [2]

$\lambda\Pi$ termes

$t\ , t' ::=$	V	variable
	$\mid \mathtt{TYPE}$	sorte des types
	$\mid \Pi\ (V : t),\ t'$	produit dépendant
	$\mid \lambda\ (V : t),\ t'$	abstraction
	$\mid t\ t'$	application

Et $t \to t'$, abbréviation pour $\Pi V : t, t'$ quand V n'appartient pas à t'.

Règles

$r ::=$	$t \hookrightarrow t'$	règles de réécriture

2. `https://lambdapi.readthedocs.io/en/latest/about.html`

Logique du premier ordre [3]

« Lambdapi is a logical framework, that is, it does not come with a pre-defined logic. Instead, one has to start defining its own logic. »

Propositions

```
constant symbol Prop : TYPE;
// Associe le type d'une preuve à une proposition
injective symbol π : Prop → TYPE;
```

Datatypes

```
constant symbol Set : TYPE;
// associe un type à un datatype
injective symbol τ : Set → TYPE;
```

3. Standard library : `https://github.com/Deducteam/lambdapi-stdlib`

Logique du premier ordre

« Lambdapi is a logical framework, that is, it does not come with a pre-defined logic. Instead, one has to start defining its own logic. »

Conjonction

```
constant symbol ∧ : Prop → Prop → Prop;
notation ∧ infix left 7;
constant symbol ∧ᵢ p q: π p → π q → π (p ∧ q);
symbol ∧ₑ₁ p q : π (p ∧ q) → π p;
symbol ∧ₑ₂ p q : π (p ∧ q) → π q;
```

Implication (dans le style de Coq)

```
constant symbol ⇒ : Prop → Prop → Prop;
notation ⇒ infix right 5;
rule π ($p ⇒ $q) ↪ π $p → π $q;
```

Séquents correspondants pour la conjonction

$$\frac{\Gamma \vdash p \quad \Gamma \vdash q}{\Gamma \vdash p \wedge q} \; (\wedge_i)$$

$$\frac{\Gamma \vdash p \wedge q}{\Gamma \vdash p} \; (\wedge_{e1})$$

$$\frac{\Gamma \vdash p \wedge q}{\Gamma \vdash q} \; (\wedge_{e2})$$

ACADIE

Théorie ensembliste d'Event-B

- Théorie des types d'Event-B
  - Types prédéfinis BOOL, Z
  - Types déclarés par l'utilisateur
- Opérateurs usuels sur les ensembles, ensembles en compréhension
- Relations et opérateurs relationnels
- Fonctions définies à partir des relations.

Théorie des ensembles d'Event-B

Types d'Event-B :

$T ::=$	$\sigma\mathbb{P}\ T$	ensemble des parties
	$\mid\ T\ \sigma\times\ \ T$	produit cartésien
	$\mid\ \sigma\mathrm{BOOL}\ \mid\ \sigma\mathbb{Z}$	booléens et entiers
	$\mid\ \ \sigma U$	pour chaque datatype utilisateur U

Traduction lambdapi :

```
injective symbol σℙ: Set → Set; // ensemble des parties
injective symbol σ×: Set → Set → Set; // produit cartésien
notation σ× infix left 24;
constant symbol σBOOL: Set; // ensemble prédifini
constant symbol σℤ: Set; // ensemble prédéfini
constant symbol σS: Set; // pour chaque ensemble utilisateur
```

ACADIE

Opérateurs sur les ensembles

Les opérateurs sur les ensembles dérivent de l'opérateur d'appartenance :

```
symbol ∈ [T:Set] : τ T → τ (σℙ T) → Prop;
symbol ⊆ [T] (s1:τℙ T) (s2: τℙ T): Prop;
constant symbol ∩ [T:Set] : τℙ T → τℙ T → τℙ T;
rule $s1 ⊆ $s2 ↪ `∀ x, x ∈ $s1 ⇒ x ∈ $s2;
rule $x ∈ $s1 ∩ $s2 ↪ $x ∈ $s1 ∧ $x ∈ $s2;
```

Relations et fonctions

```
symbol ↔ [T1 T2:Set] (A:τℙ T1) (B: τℙ T2): τℙ (σℙ (T1 σ× T2))≔ℙ (A × B);
constant symbol ⇸ [T1:Set] [T2:Set]: τℙ T1 → τℙ T2 → τℙ (σℙ (T1 σ× T2));
rule $f ∈ ($A ⇸ $B) ↪ $f ∈ ($A ↔ $B) ∧
    (`∀ x, `∀ y1, `∀ y2, x ↦ y1 ∈ $f ∧ x ↦ y2 ∈ $f ⇒ y1 = y2);
constant symbol → [T1:Set] [T2:Set]: τℙ T1 → τℙ T2 → τℙ (σℙ (T1 σ× T2));
rule $f ∈ $A → $B ↪ dom $f = $A ∧ $f ∈ $A ⇸ $B;
```

Preuves dans Rodin

Théorème de Cantor

Dans Rodin

```
cantor
context cantor

sets S

axioms
  theorem @th ¬(∃f·f∈S↠ℙ(S))
end
```

f surjection de S dans $\mathbb{P}((S))$, se traduit en Event-B par :
$\boxed{f \in \mathbb{P}(S \times \mathbb{P}((S)))}$ et
f est une fonction totale et
f une surjection.

Dans Lambdapi

```
constant symbol σS: Set;
symbol S: τ (σℙ σS) ≔ BIG;
symbol th:π(¬((`∃ (f: τ(σℙ (σS σ× (σℙ σS)))), f ∈ (S↠(ℙ S))))) ≔
... end;
```

Transformation des nœuds de l'arbre de preuve

Règle Rodin	Lambdapi	Règle Rodin	Lambdapi
$\dfrac{}{\Gamma, p \vdash p}$ (hyp)	refine H_p	$\dfrac{\Gamma, x_i \in T_i \vdash p}{\Gamma \vdash \forall x_1, \ldots, x_n \cdot p}$ ($\forall$goal)	assume $x_1 \ldots x_n$
$\dfrac{p, \Gamma \vdash q}{\Gamma \vdash p \Rightarrow q}$ ($\Rightarrow$ goal)	assume H_p	$\dfrac{\Gamma \vdash p \quad \Gamma \vdash q}{\Gamma \vdash p \wedge q}$ ($\wedge$goal)	apply $\wedge_i$ p q

- H : mapping hypothèses Rodin - hypothèse Lambdapi

ACADIE

Transformations avancées

- Opérateurs n-aires ($\wedge_i$ imbriqués)
- Tactiques de simplification de Rodin (simulation d'un setoïd-rewrite)
- Comportement de certaines règles non formellement spécifiées, présentes dans le code Java de Rodin
- Règles trop complexes ou prouveurs automatiques $\leadsto$ appel à ZenonModulo

Conclusion

Dedukti/Lambdapi est un framework logique basé sur le $\lambda\Pi$-calcule modulo des réécritures, qui souhaite favoriser l'interopérabilité entre méthodes formelles.

Nous avons présenté quelques étapes de notre plongement d'Event-B vers Lambdapi, permettant de traduire des énoncés et des arbres de preuve de Rodin. A partir des preuves de plusieurs énoncés du théorème de Cantor, nous avons pu soulever plusieurs difficultés et orienter notre travail.

Travail en cours

- Continuer la traduction des règles de déduction
- Maîtriser la complexité de la preuve générée
- Intégrer les résultats des prouveurs internes et externes de Rodin
- Traduire le comportement des machines et des évènements d'Event-B

Merci pour votre attention