Coq Tacticals and PVS Strategies: A Small Step Semantics

Florent Kirchner

florent.kirchner@inria.fr ; fkirchner@nianet.org

- The strategies, one framework.

- The strategies, one framework.
- No accurate documentation, need for thorough understanding.

- The strategies, one framework.
- No accurate documentation, need for thorough understanding.
- Prover interoperability.

- The strategies, one framework.
- No accurate documentation, need for thorough understanding.
- Prover interoperability.
- Coq's and PVS's common heritage: LCF theorem prover (1979).

- The strategies, one framework.
- No accurate documentation, need for thorough understanding.
- Prover interoperability.
- Coq's and PVS's common heritage: LCF theorem prover (1979).
- Small step semantics.

- Coq
  - Procedural,
  - Calculus of Inductive Constructions,
  - Intuitionnistic logic.
- PVS
  - Procedural,
  - Enriched Simple Types Theory,
  - Classical logic.

- Ancestor : LCF
  (M. Gordon, R. Milner, C. Wadsworth 1979)

- Ancestor : LCF
  (M. Gordon, R. Milner, C. Wadsworth 1979)
- Historically: exchange of concepts.

- Ancestor : LCF
  (M. Gordon, R. Milner, C. Wadsworth 1979)
- Historically: exchange of concepts.
- Common feature : proof language.

```
(then (split) (case A))
```

Intro ; Cut A.

- Ancestor : LCF
  (M. Gordon, R. Milner, C. Wadsworth 1979)
- Historically: exchange of concepts.
- Common feature : proof language.

```
(then (split) (case A))
```

  Intro ; Cut A.

- Independance w.r.t. implementation differences?

```
(spread (try (split)
             (skip)
             (then (flatten) (skip)))
        ((assert) (split) (assert)) ).
```

```
(spread (try (split)
             (skip)
             (then (flatten) (skip)))
        ((assert) (split) (assert)) ).
```

- Strategies – Tacticals: `spread`, `try` ,`then`.

```
(spread (try (split)
              (skip)
              (then (flatten) (skip)))
         ((assert) (split) (assert)) ).
```

- Strategies – Tacticals: `spread`, `try` ,`then`.
- Tactics: `split`, `skip`, `flatten`, `assert`.

```
(spread (try (split)
             (skip)
             (then (flatten) (skip)))
        ((assert) (split) (assert)) ).
```

- Strategies – Tacticals: `spread`, `try` ,`then`.
- Tactics: `split`, `skip`, `flatten`, `assert`.

$\leadsto$ What is the difference ?

```
(spread (try (split)
             (skip)
             (then (flatten) (skip)))
        ((assert) (split) (assert)) ).
```

- Strategies – Tacticals: `spread`, `try` ,`then`.
- Tactics: `split`, `skip`, `flatten`, `assert`.

⇝ What is the difference ?

⇝ How do they operate ?

$$\dfrac{\dfrac{}{C, (A \Rightarrow B), A \vdash C} \quad \dfrac{\dfrac{}{B, A \vdash B, C} \quad A \vdash A, B, C}{(A \Rightarrow B), A \vdash B, C} \quad \dfrac{}{(A \Rightarrow B), A \vdash A, C}}{\dfrac{(A \Rightarrow (B \Rightarrow C)), (A \Rightarrow B), A \vdash C}{\vdash (A \Rightarrow (B \Rightarrow C)) \wedge (A \Rightarrow B) \wedge A \Rightarrow C}}$$

$$\dfrac{\dfrac{\dfrac{}{C,(A \Rightarrow B),A \vdash C} \qquad \dfrac{\dfrac{}{B,A \vdash B,C} \quad A \vdash A,B,C}{(A \Rightarrow B),A \vdash B,C} \qquad \dfrac{}{(A \Rightarrow B),A \vdash A,C}}{(A \Rightarrow (B \Rightarrow C)),(A \Rightarrow B),A \vdash C}}{\vdash (A \Rightarrow (B \Rightarrow C)) \wedge (A \Rightarrow B) \wedge A \Rightarrow C}$$

+ goal numbering,

$$
\dfrac{\dfrac{}{C, (A \Rightarrow B), A \vdash C} \quad \dfrac{\dfrac{}{B, A \vdash B, C} \quad A \vdash A, B, C}{(A \Rightarrow B), A \vdash B, C} \quad \dfrac{}{(A \Rightarrow B), A \vdash A, C}}{\dfrac{(A \Rightarrow (B \Rightarrow C)), (A \Rightarrow B), A \vdash C}{\vdash (A \Rightarrow (B \Rightarrow C)) \wedge (A \Rightarrow B) \wedge A \Rightarrow C}}
$$

+ goal numbering,
+ proved and pendant goals

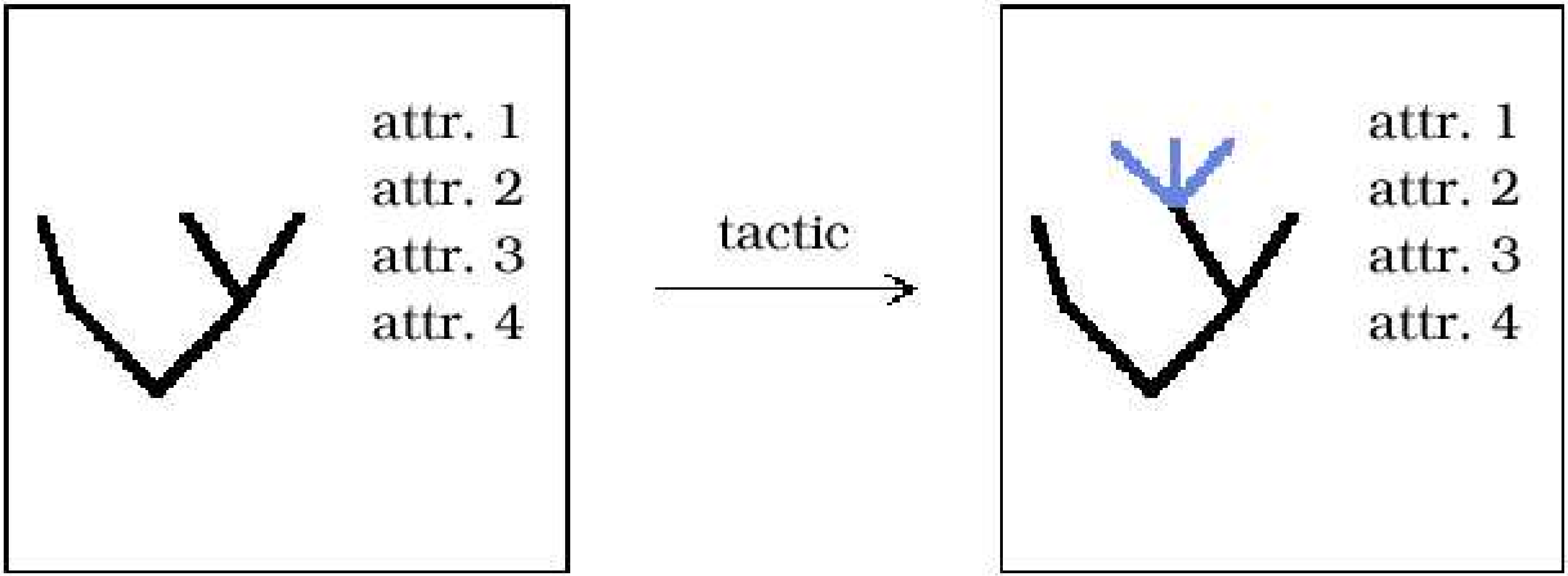
$$\dfrac{\dfrac{\dfrac{}{C, (A \Rightarrow B), A \vdash C} \qquad \dfrac{\dfrac{}{B, A \vdash B, C} \quad A \vdash A, B, C}{(A \Rightarrow B), A \vdash B, C} \qquad \dfrac{}{(A \Rightarrow B), A \vdash A, C}}{(A \Rightarrow (B \Rightarrow C)), (A \Rightarrow B), A \vdash C}}{\vdash (A \Rightarrow (B \Rightarrow C)) \wedge (A \Rightarrow B) \wedge A \Rightarrow C}$$

+ goal numbering,
+ proved and pendant goals
+ state of the proof.

$$\dfrac{\dfrac{}{C, (A \Rightarrow B), A \vdash C} \qquad \dfrac{\dfrac{}{B, A \vdash B, C} \quad A \vdash A, B, C}{(A \Rightarrow B), A \vdash B, C} \qquad \dfrac{}{(A \Rightarrow B), A \vdash A, C}}{\dfrac{(A \Rightarrow (B \Rightarrow C)), (A \Rightarrow B), A \vdash C}{\vdash (A \Rightarrow (B \Rightarrow C)) \wedge (A \Rightarrow B) \wedge A \Rightarrow C}}$$

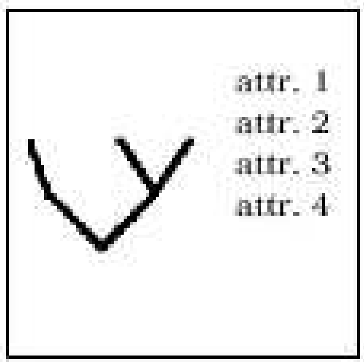
+ goal numbering,
+ proved and pendant goals
+ state of the proof.

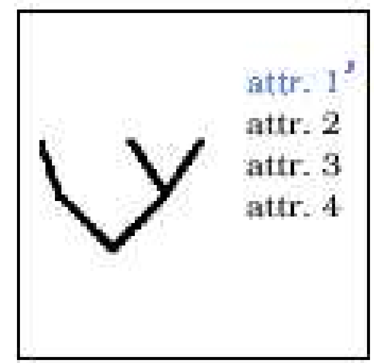
Object


attr. 1
attr. 2
attr. 3
attr. 4
tactic
attr. 1
attr. 2
attr. 3
attr. 4

TACTICALS – STRATEGIES

(if nil (fail) (split)) $\longrightarrow$ **(split)**





- Different logics...

- Different logics...
- ... but identical structures.

- Different logics...
- ... but identical structures.

- The tacticals are dependent on the tactics.

- Different logics...
- ... but identical structures.

- The tacticals are dependent on the tactics.
- Only simple tactics are independent from the implemented logic.

- Different logics…
- … but identical structures.

- The tacticals are dependent on the tactics.
- Only simple tactics are independent from the implemented logic.

$\rightsquigarrow$ Express the semantics for the tacticals, and provide some examples of simple tactics.

$\rightsquigarrow$ Expose the tacticals' small step semantics within a precise formal framework.

$\rightsquigarrow$ Implementation in Coq and PVS of new tacticals, towards a common basis.

SMALL STEP SEMANTICS

- Proof language $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_{\text{tactics}} \cup \mathcal{L}_{\text{tacticals}}$$

- Proof language $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_{\text{tactics}} \cup \mathcal{L}_{\text{tacticals}}$$

- Proof script:

$$\text{term of } \mathcal{T}(\mathcal{L}_{\text{tactiques}}, \mathcal{L}_{\text{tacticals}})$$

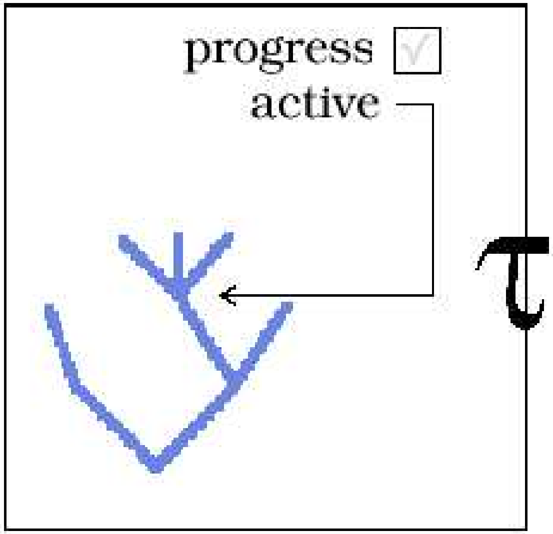
- Proof context τ: simple object.

- Proof context τ: simple object.
- Fields:

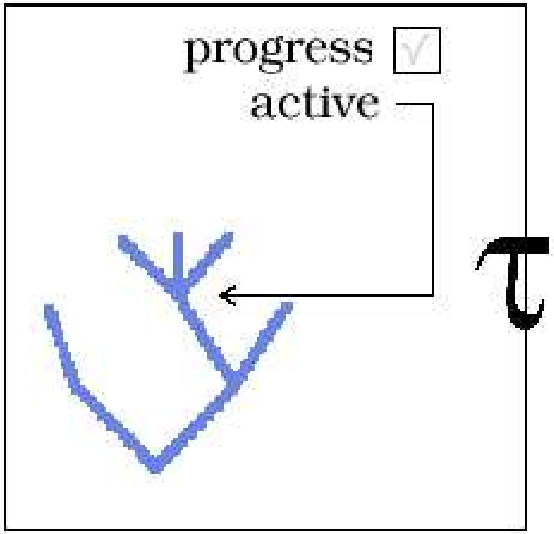
- Proof context τ: simple object.
- Fields:
  - tree of sequents,

- Proof context τ: simple object.
- Fields:
  - tree of sequents,
  - pointer on the active subtree,

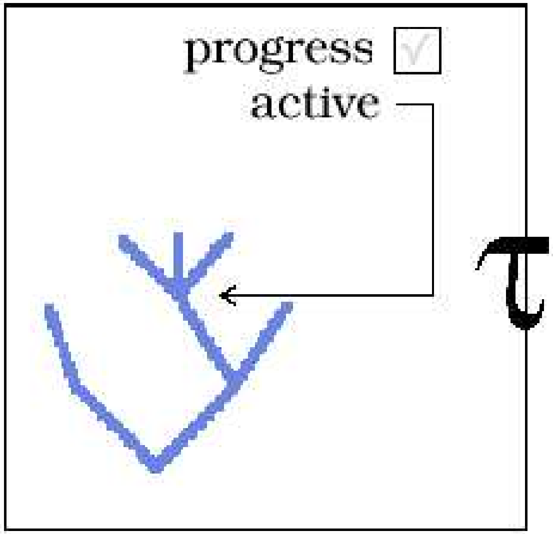
- Proof context τ: simple object.
- Fields:
  - tree of sequents,
  - pointer on the active subtree,
  - progression flag.



- Proof context τ: simple object.
- Fields:
  - tree of sequents,
  - pointer on the active subtree,
  - progression flag.
- Easy methods.



- Proof context τ: simple object.
- Fields:
  - tree of sequents,
  - pointer on the active subtree,
  - progression flag.
- Easy methods.
- Values peculiar to τ:
  $\perp_n$, $\top$, $\varnothing$.



- Tacticals' semantics parametrized by that of the tactics.

- Tacticals' semantics parametrized by that of the tactics.
- Tactic evaluation relation:

$$\text{tactic}\%_0\tau = \tau'$$

- Tacticals' semantics parametrized by that of the tactics.
- Tactic evaluation relation:

$$\text{tactic}\%_0\tau = \tau'$$

- Fonctional, total relation.

- τ. $\Gamma \vdash \Delta$ highlights the active sequent.

- $\tau.\ \ \Gamma \vdash \Delta$ highlights the active sequent.

$$\texttt{(split)}\%\tau.\ \ \Gamma \vdash A \wedge B\ = $$
$$\tau.\,\mathrm{addLeafs}\ (\Gamma \vdash A,\ \Gamma \vdash B).\,\mathrm{setProgress(true)}$$

The formalization consists in a three steps modular process which defines:

- the rewrite relation.

The formalization consists in a three steps modular process which defines:

- the rewrite relation.
- the subset of expressions that are values.

The formalization consists in a three steps modular process which defines:

- the rewrite relation.
- the subset of expressions that are values.
- the evaluation contexts that indicate where the reductions are allowed.

The formalization consists in a three steps modular process which defines:

- the rewrite relation.

$$e \;/\; \tau \xrightarrow{\epsilon} e' \;/\; \tau\text{'} \;\;,$$

- the subset of expressions that are values.
- the evaluation contexts that indicate where the reductions are allowed.

The formalization consists in a three steps modular process which defines:

- the rewrite relation.

$$e \;/\; \tau \xrightarrow{\epsilon} e' \;/\; \tau' \;\;,$$

- the subset of expressions that are values.

$$v \in \mathcal{L}_{\text{tactics}}$$

- the evaluation contexts that indicate where the reductions are allowed.

The formalization consists in a three steps modular process which defines:

- the rewrite relation.

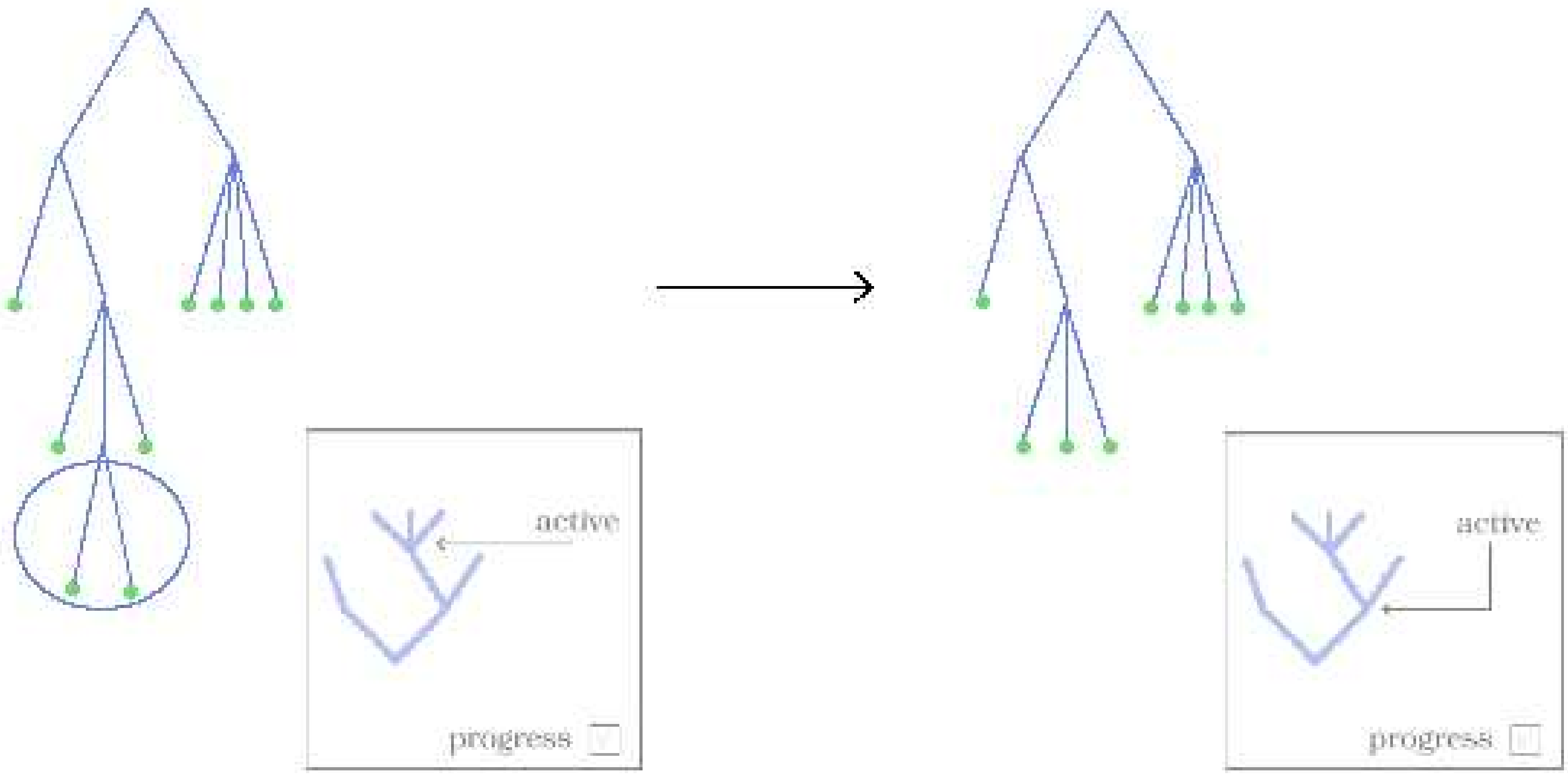
$$e \;/\; \tau \xrightarrow{\epsilon} e' \;/\; \tau\text{'} \;\;,$$

- the subset of expressions that are values.

$$v \in \mathcal{L}_{\text{tactics}}$$

- the evaluation contexts that indicate where the reductions are allowed.

$$\textit{innermost}, \textit{leftmost}\ \&\ \textit{lazy}$$


active
progress
active
progress

$$(\texttt{if}\ \ t\ \ e_1\ \ e_2)\ /\,\tau \ \xrightarrow{\ \epsilon\ }\ e_2\,/\,\tau \quad \text{if}\ t = \texttt{nil}\ .$$

$$(\texttt{if}\ \ t\ \ e_1\ \ e_2)\ /\,\tau \ \xrightarrow{\ \epsilon\ }\ e_1\,/\,\tau \quad \text{if}\ t \neq \texttt{nil}\ .$$

$$v_1 \; ; e_2 \; / \; \tau \xrightarrow{\epsilon} e_2 \; / \; (v_1 \% \tau) \qquad \textbf{if } \forall n \geq 0 \; (v_1 \% \tau) \neq \bot_n$$
$$\textbf{and } \neg (v_1 \% \tau).\,\textbf{isActiveTreeProven()} \;\; .$$

$$v_1 \; ; e_2 \; / \; \tau \xrightarrow{\epsilon} v_1 \; / \; \tau \qquad \textbf{if } \exists n \geq 0 \; (v_1 \% \tau) = \bot_n$$
$$\textbf{or } (v_1 \% \tau).\,\textbf{isActiveTreeProven()} \;\; .$$

$$\text{First } [e_1|e_2|\ldots|e_n] \;/\; \tau \quad \xrightarrow{\epsilon} \quad \overline{\text{First}_{\mathcal{T}}}\ [e_1|e_2|\ldots|e_n] \;/\; \tau$$

$$\overline{\text{First}_{\mathcal{T}}}\ [v_1|e_2|\ldots|e_n] \;/\; \tau\text{'} \quad \xrightarrow{\epsilon} \quad v_1 \;/\; \tau\text{'} \qquad \text{if } \forall n \geq 0\ (v_1 \%_0 \tau\text{'}) \neq \bot_n$$

$$\overline{\text{First}_{\mathcal{T}}}\ [v_1|e_2|\ldots|e_n] \;/\; \tau\text{'} \quad \xrightarrow{\epsilon} \quad \text{First } [e_2|\ldots|e_n] \;/\; \tau \qquad \text{if } \exists n \geq 0\ (v_1 \%_0 \tau\text{'}) = \bot_n \quad .$$

$$\overline{\text{First}_{\mathcal{T}}}\ [\,] \;/\; \tau\text{'} \quad \xrightarrow{\epsilon} \quad (\text{Fail } 0) \;/\; \tau \quad .$$

$$(\texttt{try}\ e_1\ e_2\ e_3)\ /\ \tau \xrightarrow{\epsilon} (\overline{\textbf{try}}_{\mathcal{T}}\ e_1\ e_2\ e_3)\ /\ \tau$$

$$(\overline{\textbf{try}}_{\mathcal{T}}\ v_1\ e_2\ e_3)\ /\ \tau\text{'} \xrightarrow{\epsilon} (\overline{\textbf{try}}_{\mathcal{T}}\ e_2)\ /\ (v_1\%\tau\text{'})\ \textbf{if}\ (v_1\%\tau\text{'}).\textbf{hasProgressed}$$
$$\textbf{and}\ \forall n \geq 0\ (v_1\%\tau\text{'}) \neq \bot_n$$

$$(\overline{\textbf{try}}_{\mathcal{T}}\ v_1\ e_2\ e_3)\ /\ \tau\text{'} \xrightarrow{\epsilon} v_1\ /\ \tau \quad \textbf{if}\ \exists n \geq 0\ (v_1\%\tau) = \bot_n\ .$$

$$(\overline{\textbf{try}}_{\mathcal{T}}\ v_1\ e_2\ e_3)\ /\ \tau\text{'} \xrightarrow{\epsilon} e_3\ /\ \tau\text{'} \quad \textbf{if}\ \neg(v_1\%\tau).\textbf{hasProgressed()}\ .$$

with

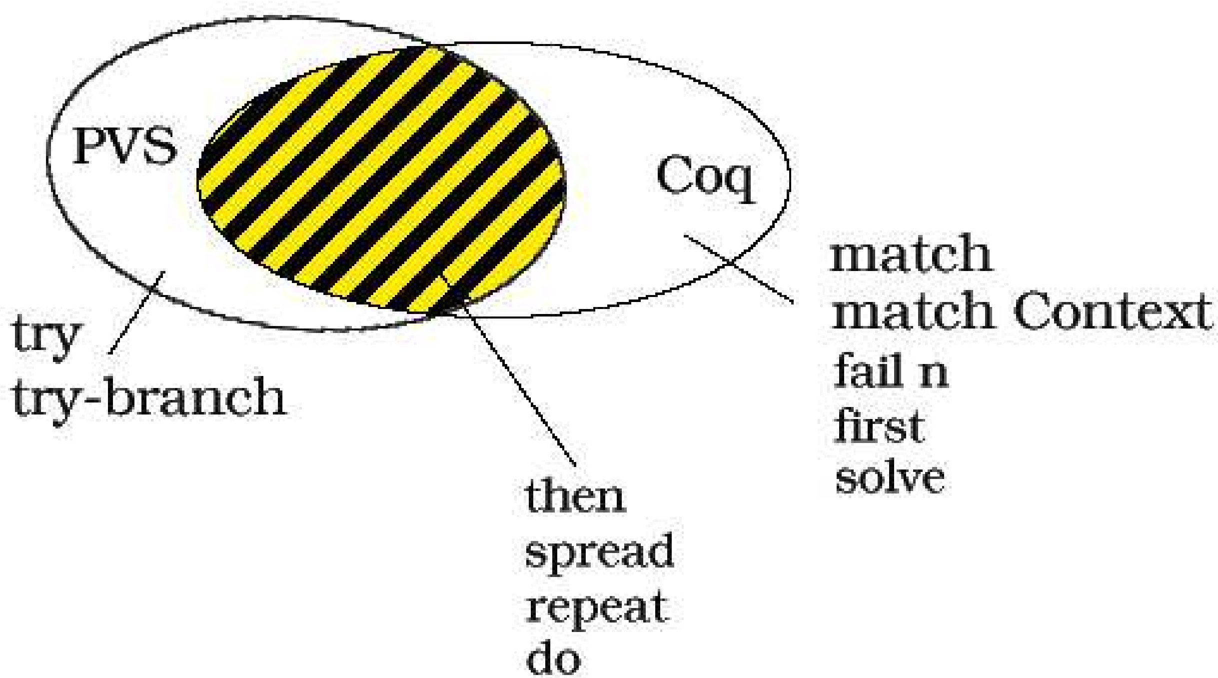
$$(\overline{\textbf{try}}_{\mathcal{T}}\ v)\ /\ \tau\text{'} \xrightarrow{\epsilon} v\ /\ \tau\text{'} \quad \textbf{if}\ \forall n \geq 0\ (v\%\tau\text{'}) \neq \bot_n\ .$$

$$(\overline{\textbf{try}}_{\mathcal{T}}\ v)\ /\ \tau\text{'} \xrightarrow{\epsilon} \overline{\textbf{bt}}\ /\ \tau \quad \textbf{if}\ \exists n \geq 0\ (v\%\tau\text{'}) = \bot_n\ .$$

- Expression of all Coq tacticals and most significant PVS strategies.

- Expression of all Coq tacticals and most significant PVS strategies.
- Integrity through formalization.

- Expression of all Coq tacticals and most significant PVS strategies.
- Integrity through formalization.
- Allows for a comparison between the two languages.



CARRYING OUT

In PVS:

- match, match context

In PVS:

- match, match context
- named errors, catch

In PVS:

- match, match context
- named errors, catch
- first, solve

In PVS:

- match, match context
- named errors, catch
- first, solve
- miscellaneous: for, when, while, etc.

In PVS:

- match, match context
- named errors, catch
- first, solve
- miscellaneous: for, when, while, etc.

$\rightsquigarrow$ Towards a coherent programming language

In Coq:

- try

In Coq:

- try
- try-branch

In Coq:

- try
- try-branch
- miscellaneous: Case-Test.

In Coq:

- try
- try-branch
- miscellaneous: Case-Test.

$\leadsto$ Introduction of sophisticated tacticals.

- Enhancement of the tactical interoperability.

- Enhancement of the tactical interoperability.
- Valuable mutual enrichment .

- Enhancement of the tactical interoperability.
- Valuable mutual enrichment .
- Documentation of new functionalities.

- Enhancement of the tactical interoperability.
- Valuable mutual enrichment .
- Documentation of new functionalities.
- Real world testing conditions:

`http://research.nianet.org/fm-at-nia/Practicals/`

- Good basis concerning proof language interoperability.

- Good basis concerning proof language interoperability.
- Deepening: formalism reevaluation, tactics' semantics.

- Good basis concerning proof language interoperability.
- Deepening: formalism reevaluation, tactics' semantics.
- Widening: Nurpl, Isabelle, ELAN.

- Good basis concerning proof language interoperability.
- Deepening: formalism reevaluation, tactics' semantics.
- Widening: Nurpl, Isabelle, ELAN.
- Continuation of the teams' cooperation.

- Good basis concerning proof language interoperability.
- Deepening: formalism reevaluation, tactics' semantics.
- Widening: Nurpl, Isabelle, ELAN.
- Continuation of the teams' cooperation.
- Source code maintenance and improvement.

- Gilles Dowek and César Muñoz.
- Alfons Geser, Assia Mahboubi.
- NIA and NASA formal method groups, LogiCal team.
- ENS Cachan, NIA and INRIA.