Extensible Proof Decomposition Rules for TLAPS

Karolis Petrauskas

Vilnius University

TLA$^+$ Community Event
2026-04-12

Introduction

Proof decomposition refers proof steps,
that are provided by a user,
but are mostly mechanical.

Example

Having

```
...
⟨1⟩1. SUFFICES ASSUME Inv, [Next]_vars PROVE Inv' BY ...
⟨1⟩q. QED BY DEF Next
```

Example

Having

```
...
⟨1⟩1. SUFFICES ASSUME Inv, [Next]_vars PROVE Inv' BY ...
⟨1⟩q. QED BY DEF Next
```

The proof obligation at QED is

```
ASSUME
    Inv,
    ActionA ∨ ActionB
PROVE Inv'
```

Example

Having

```
...
⟨1⟩1. SUFFICES ASSUME Inv, [Next]_vars PROVE Inv' BY ...
⟨1⟩q. QED BY DEF Next
```

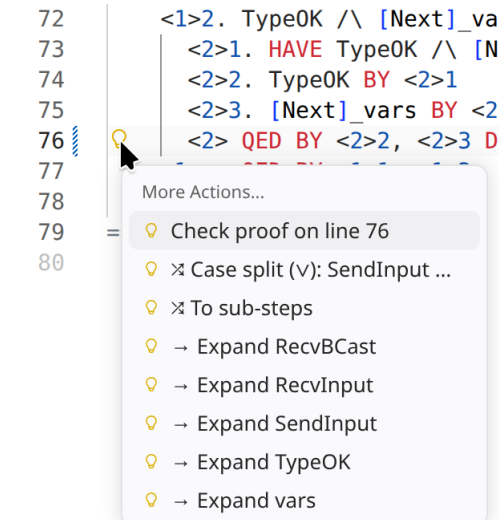
The proof obligation at QED is

```
ASSUME
    Inv,
    ActionA ∨ ActionB
PROVE Inv'
```

We get

```
...
⟨1⟩1. SUFFICES ASSUME Inv, [Next]_vars PROVE Inv' BY ...
⟨1⟩a. CASE ActionA
⟨1⟩b. CASE ActionB
⟨1⟩q. QED BY ⟨1⟩a, ⟨1⟩b DEF Next
```

Existing Tool



- Implemented in the TLAPM LSP, as code actions.
- Covers $\Rightarrow, \vee, \wedge, \forall, \exists, \equiv$, for a goal and for assumptions.
- Also, allows transform a step to nested, and to expand a definition.
- Tries to keep the decision on nesting to the user.

Decomposition rules for sets, functions,

Decomposition rules for sets, functions, ...
and user defined structures?

**Decomposition rules for sets, functions, ...
and user defined structures?**

- Implementing them into the LSP sounds cumbersome.
  And cannot handle decompositions based on user definitions.

Decomposition rules for sets, functions, ... and user defined structures?

- Implementing them into the LSP sounds cumbersome.
  And cannot handle decompositions based on user definitions.
- How to encode the rules to be used by the LSP?
  Better avoid inventing another syntax.

Encoding a Rule

Encoding

In the previous example

```
...
⟨1⟩1. SUFFICES ASSUME Inv, [Next]_vars PROVE Inv' BY ...
⟨1⟩a. CASE ActionA
⟨1⟩b. CASE ActionB
⟨1⟩q. QED BY ⟨1⟩a, ⟨1⟩b DEF Next
```

The rule applied is

$$\frac{A \vdash Q \qquad B \vdash Q}{A \vee B \vdash Q} \quad (\mathrm{L}\vee)$$

Encoding

$$\frac{A \vdash Q \qquad B \vdash Q}{A \lor B \vdash Q} \quad (\mathrm{L}\lor)$$

```
1  THEOREM
2      ASSUME
3          NEW A, NEW B, NEW Q,
4          A ∨ B
5      PROVE Q
6  PROOF
7      ⟨1⟩1. ASSUME A PROVE Q
8      ⟨1⟩2. ASSUME B PROVE Q
9      ⟨1⟩q. QED BY ⟨1⟩1, ⟨1⟩2
```

We map it as follows:

- The conclusion becomes the theorem statement.
- The premises become the proof steps.

Such encoding also allows to check the validity of the rule by the TLAPS itself.

Encoding

$$\frac{A \vdash Q \qquad B \vdash Q}{A \lor B \vdash Q} \quad (\mathrm{L}\lor)$$

```
1  THEOREM
2      ASSUME
3          NEW A, NEW B, NEW Q,
4          A ∨ B
5      PROVE Q
6  PROOF
7      ⟨1⟩1. ASSUME A PROVE Q
8      ⟨1⟩2. ASSUME B PROVE Q
9      ⟨1⟩q. QED BY ⟨1⟩1, ⟨1⟩2
```

Such encoding also allows to check the validity of the rule by the TLAPS itself.

We map it as follows:

- The conclusion becomes the theorem statement.
- The premises become the proof steps.

The proof obligation at ⟨1⟩q:

```
ASSUME ...
    ASSUME A PROVE Q,
    ASSUME B PROVE Q,
    A ∨ B
PROVE Q
```

Identifying a Rule

How to Mark Theorem as a Rule

- Annotations. Requires changes to the current parsers.

How to Mark Theorem as a Rule

- Annotations. Requires changes to the current parsers.
- Theorem Naming. We don't want to use rules as facts.

How to Mark Theorem as a Rule

- Annotations. Requires changes to the current parsers.
- Theorem Naming. We don't want to use rules as facts.
- Introduce a fresh variable with a specific name.

```
1  THEOREM
2      ASSUME NEW DECOMPOSITION_RULE,
3          NEW A, NEW B, NEW Q,
4          A ∨ B
5      PROVE Q
6  PROOF
7      ⟨1⟩1. ASSUME A PROVE Q
8      ⟨1⟩2. ASSUME B PROVE Q
9      ⟨1⟩q. QED BY ⟨1⟩1, ⟨1⟩2
```

Which Rule to Apply?

Pattern Matching

The context

```
ASSUME
    Inv,
    ActionA ∨ ActionB
PROVE  Inv′
```

matches pattern

```
ASSUME NEW DECOMPOSITION_RULE,
    NEW A, NEW B, NEW Q,
    A ∨ B
PROVE  Q
```

Pattern Matching

The context

```
ASSUME
    Inv,
    ActionA ∨ ActionB
PROVE  Inv′
```

matches pattern

```
ASSUME NEW DECOMPOSITION_RULE,
    NEW A, NEW B, NEW Q,
    A ∨ B
PROVE  Q
```

with substitution

$$\{ \quad A \mapsto \text{ActionA}, \quad B \mapsto \text{ActionB}, \quad Q \mapsto \text{Inv}' \quad \}.$$

Pattern Matching

The context

```
ASSUME
    Inv,
    ActionA ∨ ActionB
PROVE  Inv′
```

matches pattern

```
ASSUME NEW DECOMPOSITION_RULE,
    NEW A, NEW B, NEW Q,
    A ∨ B
PROVE  Q
```

with substitution

$$\{ \quad A \mapsto \text{ActionA}, \quad B \mapsto \text{ActionB}, \quad Q \mapsto \text{Inv}' \quad \}.$$

- The fresh variables are considered placehoders.
- Order of assumptions is ignored.

How to Apply the Rule?

How to Apply the Rule

```
⟨1⟩1. SUFFICES ASSUME Inv, [Next]_vars PROVE Inv' BY ...
⟨1⟩q. QED BY DEF Next
```

After the rule is found and matched, we

- Apply the substitution in the rule's proof.
- Rename the proof steps.
- Merge context QED proof with the rule's QED proof.
- Propose code action replacing old QED text with the transformed rule proof.

```
⟨1⟩1. ASSUME A PROVE Q
⟨1⟩2. ASSUME B PROVE Q
⟨1⟩q. QED BY ⟨1⟩1, ⟨1⟩2
```

```
⟨1⟩2. ASSUME ActionA PROVE Inv'
⟨1⟩3. ASSUME ActionB PROVE Inv'
⟨1⟩q. QED BY ⟨1⟩2, ⟨1⟩3 DEF Next
```

Requirements

The rule has to match the following requirements:

- Introduces fresh variable `DECOMPOSITION_RULE`.
- Is a structured proof. Has 1 or more levels.
- The QED has no proof, or a leaf proof.
- Can refer to the global definitions.

The context has:

- Matches the rule's theorem header.
- Has no proof or a leaf proof.

Examples

Examples

Replace goal $x \in A \cap B$ with $x \in A$ and $x \in B$.

```
1  THEOREM
2      ASSUME NEW DECOMPOSITION_RULE, NEW x, NEW A, NEW B
3      PROVE x ∈ A ∩ B
4  PROOF
5      ⟨1⟩1. x ∈ A
6      ⟨1⟩2. x ∈ B
7      ⟨1⟩q. QED BY ⟨1⟩1, ⟨1⟩2
```

Examples

Introduce $\text{PICK } y \in S : x = f(y)$
for assumption $x \in \{f(y) : y \in S\}$.

```
1  THEOREM
2      ASSUME NEW DECOMPOSITION_RULE,
3          NEW x, NEW f(_), NEW S, NEW Q
4          x ∈ {f(y) : y ∈ S}
5      PROVE Q
6  PROOF
7      ⟨1⟩1. PICK y ∈ S : x = f(y)
8      ⟨1⟩q. QED BY ⟨1⟩1
```

Discussion

- The rules are expressed in TLA$^+$, but have to follow conventions.

Discussion

- The rules are expressed in TLA$^+$, but have to follow conventions.
- The rules don't need to be global, but instead defined in separate modules and included as needed.

Discussion

- The rules are expressed in TLA$^+$, but have to follow conventions.
- The rules don't need to be global, but instead defined in separate modules and included as needed.
- Pattern matching can be used to find related theorems.