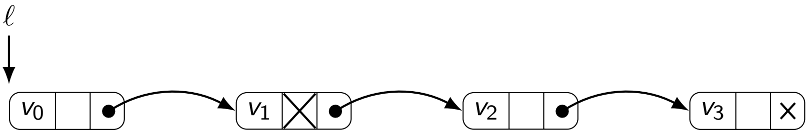
Extending the Iris Proof Mode with Inductive Predicates using Elpi

Luko van der Maas

Computing Science
Radboud University

The goal



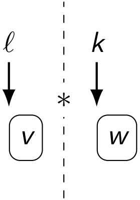
Coq

```
eiInd
Inductive is_MLL : val → list val → iProp :=
    | empty_is_MLL : is_MLL NONEV []
    | mark_is_MLL v vs l tl :
      l ↦ (v, #true, tl) -∗ is_MLL tl vs -∗
      is_MLL (SOMEV #l) vs
    | cons_is_MLL v vs tl l :
      l ↦ (v, #false, tl) -∗ is_MLL tl vs -∗
      is_MLL (SOMEV #l) (v :: vs).
```

Program verification

- Verify programs by specifying pre and post conditions
- Specification happens in separation logic

Separation logic



$$\ell \mapsto v \;*\; k \mapsto w$$

$$[\text{isMLL}\ hd\ \vec{v}]\ \text{delete}\ \ hd\ i\,[\text{isMLL}\ hd\ (\text{remove}\ i\ \vec{v})]$$

Representation predicates

$$\mathrm{isMLL}\ hd\ \vec{v} = \begin{array}{l} (hd = \mathbf{none} * \vec{v} = []) \vee \\ (\exists \ell, v', tl.\ hd = \mathbf{some}\ l * l \mapsto (v', \mathbf{true}, tl) * \mathrm{isMLL}\ tl\ \vec{v}) \vee \\ \left(\begin{array}{c} \exists \ell, v', \vec{v}'', tl.\ hd = \mathbf{some}\ l * l \mapsto (v', \mathbf{false}, tl) * \\ \vec{v} = v' :: \vec{v}'' * \mathrm{isMLL}\ tl\ \vec{v}'' \end{array} \right) \end{array}$$

isMLL-IND

$$\frac{\begin{array}{c} \mathrm{True} \vdash \Phi\ \mathbf{none}\ [] \qquad \ell \mapsto (v', \mathbf{true}, tl) * (\mathrm{isMLL}\ tl\ \vec{v} \wedge \Phi\ tl\ \vec{v}) \vdash \Phi\,(\mathbf{some}\ \ell)\ \vec{v} \\ \ell \mapsto (v', \mathbf{false}, tl) * (\mathrm{isMLL}\ tl\ \vec{v} \wedge \Phi\ tl\ \vec{v}) \vdash \Phi\,(\mathbf{some}\ \ell)\,(v' :: \vec{v}) \end{array}}{\mathrm{isMLL}\ hd\ \vec{v} \vdash \Phi\ hd\ \vec{v}}$$

Problem

- isMLL can't just be defined because Coq
- ...

Inductive command

Coq

```
eiInd
Inductive is_MLL : val → list val → iProp :=
    | empty_is_MLL : is_MLL NONEV []
    | mark_is_MLL v vs l tl :
      l ↦ (v, #true, tl) -∗ is_MLL tl vs -∗
      is_MLL (SOMEV #l) vs
    | cons_is_MLL v vs tl l :
      l ↦ (v, #false, tl) -∗ is_MLL tl vs -∗
      is_MLL (SOMEV #l) (v :: vs).
```

Tactics

- `eiInduction`
- `eiIntros` & `eiDestruct`
- Lemmas: `empty_is_MLL`, `mark_is_MLL`, `cons_is_MLL`

Demo

Demo of delete on $is_M LL$

Outline of construction

- Transform inductive definition into a pre fixpoint function
- Prove that pre fixpoint function is monontone
- Take least fixpoint of pre fixpoint function

Pre fixpoint function

Least fixpoint

Elpi

- λProlog dialect
- example
- ...

Coq-Elpi

Implementing tactics

Structure of `eiInd`

Contributions

- Created system for defining and using inductive predicates in the IPM
- Strategy for modular tactics in Elpi
- Generating monotonicity proof of pre fixpoint function
- Evaluation of Elpi as meta-programming language for the IPM