Extending the Iris Proof Mode with Inductive Predicates using Elpi

Luko van der Maas

Computing Science
Radboud University

Program verification

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

Separation logic with Hoare triples

Representation predicate

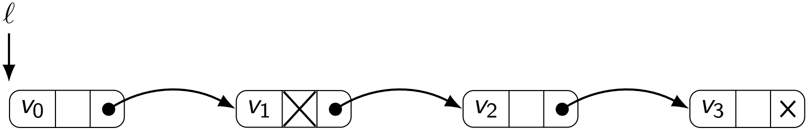
$[\mathrm{isD}\ d\ y]\ \mathrm{op}\ d\ x\ [\mathrm{isD}\ d\ (\mathrm{f}\ x\ y)]$

Imperative program

Functional program

$[\mathrm{isList}\ hd\ \vec{v}]\ \mathrm{delete}\ \ hd\ i\ [\mathrm{isList}\ hd\ (\mathrm{remove}\ i\ \vec{v})]$

Representation predicates



$$
\begin{aligned}
\text{isMLL}\ hd\ \vec{v} = \;& (hd = \textbf{none} * \vec{v} = []) \lor \\
& (\exists \ell, v', tl.\ hd = \textbf{some}\ l * l \mapsto (v', \texttt{true}, tl) * \text{isMLL}\ tl\ \vec{v}) \lor \\
& \left(\begin{array}{c} \exists \ell, v', \vec{v}'', tl.\ hd = \textbf{some}\ l * l \mapsto (v', \texttt{false}, tl) * \\ \vec{v} = v' :: \vec{v}'' * \text{isMLL}\ tl\ \vec{v}'' \end{array} \right)
\end{aligned}
$$

Outline of our solution

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).
```

Approach

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

Demo

Conclusion

Future work