Abstract Data Types

Abstract Data Types

- PVS provides support to define recursive types (list, stack, tree, syntax, etc)
- From the prelude:

```
list [T: TYPE]: DATATYPE
 BEGIN
  null: null?
  cons (car: T, cdr: list) :cons?
 END list
```

- Given as a collection of constructors, recognizers, and accessors

Abstract Data Types

Pattern matching for ADTs

- PVS provides support for a simple form of pattern-matching

-
  ```
  length(l): RECURSIVE nat =
   CASES l OF
     null: 0,
     cons(x, y): length(y) + 1
   ENDCASES
  MEASURE length(l)
  ```
- An ELSE statement can be present if not all constructors were mentioned
    - If some case is missing, a specific TCC is generated on typechecking

Abstract Data Types

Implicit declarations

During type checking an ADT, PVS automatically generates:

- Definitions for the type, constructors, recognizers, and accessors
  - As uninterpreted declarations
- Axioms providing meaning
- Additional operators
  - subterm, <<, reduce_nat, reduce_ordinal

Abstract Data Types

Implicit declarations - Axioms

- Extensionality:
  - there is only one bottom element for every constant constructor and
  - elements are distinguishable by the accessors

- Eta axiom: if the values returned by accessors are used to construct a new element, the same element is constructed

- Meaning of accessors

- Inclusion and Disjointness: recognizers characterize all the elements

- Structural Induction Scheme

Advanced Tips

Use Judgements to Avoid TCC Explosion

- TCCs appears also as obligations during a proof
- Judgements can be use to add information for the type checker
- This information is used also on proving sessions

Use Recursive Judgements to Avoid Induction

- Recursive Judgements applies only on recursive definitions
- Follow each recursive call
- More restricted than general judgements
    - Can be stated only on applications

Use Higher-Order to Simulate Mutual Recursion

- Mutual recursion is not natively supported by PVS
- But there are ways to make it happen…

Advanced Tips

- Use judgements to avoid TCC explosions (and hence proof-step repetitions)
- Use recursive judgements to simplify induction proofs
- Use higher order to simulate mutual recursion
- Use the types to collect valuable information
- Use strategies for meta-logic manipulation

Where can I learn more on PVS?

Resources

- “Applied Logic for Computer Scientists”
  - by Mauricio Ayala & Flavio de Moura
- Manuals at PVS website:
  - https://pvs.csl.sri.com/documentation.html (also locally at <PVS dir>/doc/)
- PVS google group:
  - https://groups.google.com/g/pvs-group
- Write to mariano.m.moscato@nasa.gov

Where can I learn more on PVS?

Tutorial, Classes, Courses, etc.

- Tutorial at CADE 2021:
  - https://shemesh.larc.nasa.gov/fm/pvs/TutorialCADE2021/
- PVS Class at ITP 2017:
  - http://www.mat.unb.br/ayala/pvsclass17/index.html
- Class at NASA 2012:
  - https://shemesh.larc.nasa.gov/PVSClass2012/schedule.html