Modularity

- Specifications are modularized as *collections of libraries*
- Libraries group *theories*
- Theories contain *declarations*

Declarations

- Types
  - <identifier> : **type**[**+**] [= <definition>]
- Functions/predicates/constants
  - <identifier> : <type> [= <definition>]
- Formulas
  - <identifier> : <kind> <definition>
  - Assumed valid: axiom or postulate, Proof obligation: theorem, lemma, etc.

Types

- Collection of elements, possibly empty, possibly infinite
- Type expressions: real, [int -> real], [real, real], [# x: real, y: real #], …
  - Allow to define these collections
- Types do not need to have a name
  - But remember that (formal) specifications are intended for persons (too)
- PVS uses structural equivalence not name equivalence
  - The *structure* of two types is what define equivalence between them

Types

- Uninterpreted: no assumptions (beside non-emptiness)
  - T: type
  - number: type+
- Subtyping
  - number_field: type+ from number
  - real, $\mathbb{R}$: type+ from number_field
  - rational, rat, $\mathbb{Q}$: type+ from real

Constant Declarations

Including Functions, Predicates, Relations, and *0-ary* Constants

- <identifier> : <type> [= <definition>]
- Function for evaluating a degree 2 polynomial with coefficients a, b, and c
  - eval: [real, real, real, real -> real] = λ(a, b, c, x: real): a*x^2 + b*x + c
  - eval(a, b, c: real, x: real): real = a*x^2 + b*x + c
  - eval(a, b, c: real)(x: real): real = a*x^2 + b*x + c

Type Declarations (2)
Predicate Subtyping

- The variables *a, b, c* represent coefficients of a quadratic polynomial
- Then *a* should never be zero
  - eval(a: **nzreal**, b, c: real)(x: real): real = a*x^2 + b*x + c
- From the prelude
  - nzreal, nonzero_real: NONEMPTY_TYPE = {r: real | r /= 0} CONTAINING 1
  - nonneg_real: NONEMPTY_TYPE = {x: real | x >= 0} CONTAINING 0
  - posreal: NONEMPTY_TYPE = {x: nonneg_real | x > 0} CONTAINING 1
  - […]

Declaring Formulas

<identifier> : <kind> <definition>

- discr_symm : lemma
- ∀(a: nzreal, b, c: real): discr(a,b,c) = discr(-a,-b,-c)

Proving in PVS

The PVS prover implements a Sequent Calculus

- The prover maintains a *proof tree*, each node is a *sequent*
- Sequent: pair of collections of formulas
- Objective: construct a complete proof tree (all leaves recognized as valid)
  - Valid sequents:
- The proof starts with the sequent
- The tree grows by applying a proof step on a leave

Proving in PVS
PVS Sequents

- Intuitive meaning of sequents

- Some equivalences

↔

PVS avoids top-level negations
(move formula to the other side)

Universal strength quantifications

Survival Guide

Commands: parenthesis, double quotes

- (skeep): skolemize universal quants
- (expand “<constant name>”)
- (lemma “<formula name>”)
- (inst <form num> “<expr>” … “<expr>”)
- (show-parens)
- (help “<command name>”)

M- (Meta key)	Mac: *option* key Linux: *alt* key
C-	*Control* key
Prove Formula	M-x pr C-x C-p
View Prelude	M-x view-prelude-file M-x vpf
Search Search RegExp	C-s M-C-s

More Powerful Commands

- (prop) -> propositional simplification
- (bddsimp) -> propositional simplification with Binary Decision Diagrams
- (assert) -> applies type-specific decision procedures and auto rewrites
- (ground) -> prop + assert
- (smash) -> Repeatedly tries bddsimp, assert, and lift-if
  - (grind) -> All of the above + expand & inst?

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
- Send Me a word! — mariano.m.moscato@nasa.gov

Where can I learn more on PVS?

Tutorial, Classes, Courses, etc.

- This Friday 9:00 AM!
  - More advanced topics on specification and proving in PVS
- 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 2021:
  - https://shemesh.larc.nasa.gov/PVSClass2012/schedule.html