The Module System of PVS

Lecture Interactive Proof Tools
Summer Term 2003

Hans de Nivelle, Patrick Maier

Why Modules?

- Structuring specifications: Related parts of a large specification should be packaged together into units of managable size, cf. modules/packages/classes in programming languages.
- Abstraction through genericity: Certain types (or constants) in a specification actually stand for a class of types (or constants). This should be made explicit to prevent use of implicit assumptions.
- Reusing specifications: Instantiate generic parts of a specification by providing concrete types (or constants) for the abstract generic ones.

The Modules of PVS: Theories

A **theory** consists of

- a *name* for referencing,
- *formal parameters* for declaring generic types and constants,
- an *export clause* for hiding parts of the theory which are irrelevant to the outside world,
- *assumptions* that constrain the formal parameters,
- *import declarations* for importing other theories or instances of other theories, and
- all the usual *theory declarations*.

Theories: Names

The **name** of a theory is just an identifier. This name is to be used

- to reference a theory when it is imported, and
- to resolve ambiguous declarations in imported theories by *qualifying* them with the theory name (plus actual parameters).

Examples:

- `name1: THEORY`
  `name2[`*formal parameters*`]: THEORY`
- `IMPORTING name1`
  `IMPORTING name2`
  `IMPORTING name2[`*actual parameters*`]`
- `a_equal_a: THEOREM name1.a = name2[`*actual parameters*`].a`

Theories: Formal Parameters

Formal parameters declare generic

- types or subtypes, and
- constants or functions.

Examples:

- `groups[G: TYPE, e: G, o: [G, G -> G], inv: [G -> G]]: THEORY`
- ```
  sorted_arrays[dom, rng: TYPE,
                (IMPORTING orders[dom])
                leq_dom: (orders[dom].total_order?),
                (IMPORTING orders[rng])
                leq_rng: (orders[rng].partial_order?)]: THEORY
  ```
  ```

Theories: Assumptions

Assumptions

- constrain (the use of) the formal parameters,
- cause TCCs upon instantiation.

Example:

```
groups[G: TYPE, e: G, o: [G, G -> G], inv: [G -> G]]: THEORY
BEGIN
  ASSUMING
    a, b, c: VAR G
    assoc: ASSUMPTION a o (b o c) = (a o b) o c
    associativity: ASSUMPTION a o (b o c) = (a o b) o c
    unit: ASSUMPTION a o e = a AND e o a = a
    inverse: ASSUMPTION a o inv(a) = e AND inv(a) o a = e
  ENDASSUMING
END groups
```

Theories: Import Declarations

An **importing clause**

- brings a theory (user defined or from the prelude) into scope and
- instantiates formal parameters, which
- may lead to TCCs justifying the assumptions.

Example: The clause `IMPORTING groups[real, 0, +, -]` leads to 3 TCCs

```
TCC1: OBLIGATION FORALL (a, b, c: real): a + (b + c) = (a + b) + c
TCC2: OBLIGATION FORALL (a: real): a + 0 = a AND 0 + a = a
TCC3: OBLIGATION FORALL (a: real): a + (-)(a) = 0 AND (-)(a) + a = 0
```

Theories: Export Clause

The **export clause** restricts what may be imported into other theories.

- All names (but variables) and imported theories may be exported.
- By default, everything is exported.
- If a name is exported then all names it depends on (for type checking) must be exported, too.
- Auxiliary constructions (used in proofs only) need not be exported.

Examples: (`a,b,c` declared, `th1,th2` imported)

- `EXPORTING a, c` *or* `EXPORTING ALL BUT b`
- `EXPORTING a, c WITH th1, th2`
- `EXPORTING a, c WITH CLOSURE`
- `EXPORTING ALL WITH ALL`

References

1. PVS Language Reference. Chapter 6 and 7.

2. Judy Crow, Sam Owre, John Rushby, Natarajan Shankar, Mandayam Srivas. A Tutorial Introduction to PVS. Presented at WIFT '95, Boca Raton, Florida, April 1995.