JA

Spec

- A checker for C# programs
  - Finds null pointers, array dereferences…
  - Checks Hoare logic specifications
    - Expressed in Java Modeling Language (JML)
- Goals:
  - Find errors
  - Prove correctness
- Developed at Microsoft Research
  - Freely available for non-commercial use

JA

Demo: Contains in Spec

JA

Data Invariants

- Data Invariant
  - A predicate that is true on every entry and exit of ADT functions
    - Not usually part of public specification
    - On function entry, you can count on the invariant because the last function left the data in a good state
    - Must verify that all ADT functions preserve invariant
  - Does not have to be true in middle of function
    - May violate invariant temporarily while updating state
- Motivation
  - Reduces duplication in pre-/post-conditions
    - e.g. sorted(s)
  - Hides representation decision from clients
    - Why should public spec say the array is sorted? That's an internal decision that shouldn't affect clients if it were to change.

JA

Demo: SimpleSet in Spec

JA

Spec#'s Limitations

- Does not check for some errors
  - Infinite loops, arithmetic overflow
  - Functional properties not stated by user
  - Non-functional properties
- May report false positives
  - Often can be solved with an extra precondition or invariant
  - Spurious warnings can also be disabled

JA

Spec #'s Tradeoffs

- Attempts to automate Hoare-logic style checking
- Benefits
  - Easier than manual proof
  - Sound
  - Automatically checked
- Drawbacks
  - Still quite labor-intensive
  - Builds in a particular methodology
    - Not all code may match it
- Applicability
  - Checking of critical code
    - When it's worth the extra effort to get it right
    - When you can't do a complete Hoare-logic proof
  - Still must use other analysis techniques
    - The spec, design, quality attributes must also be validated!