High-Assurance Java Card

Alessandro Coglio

Kestrel Institute

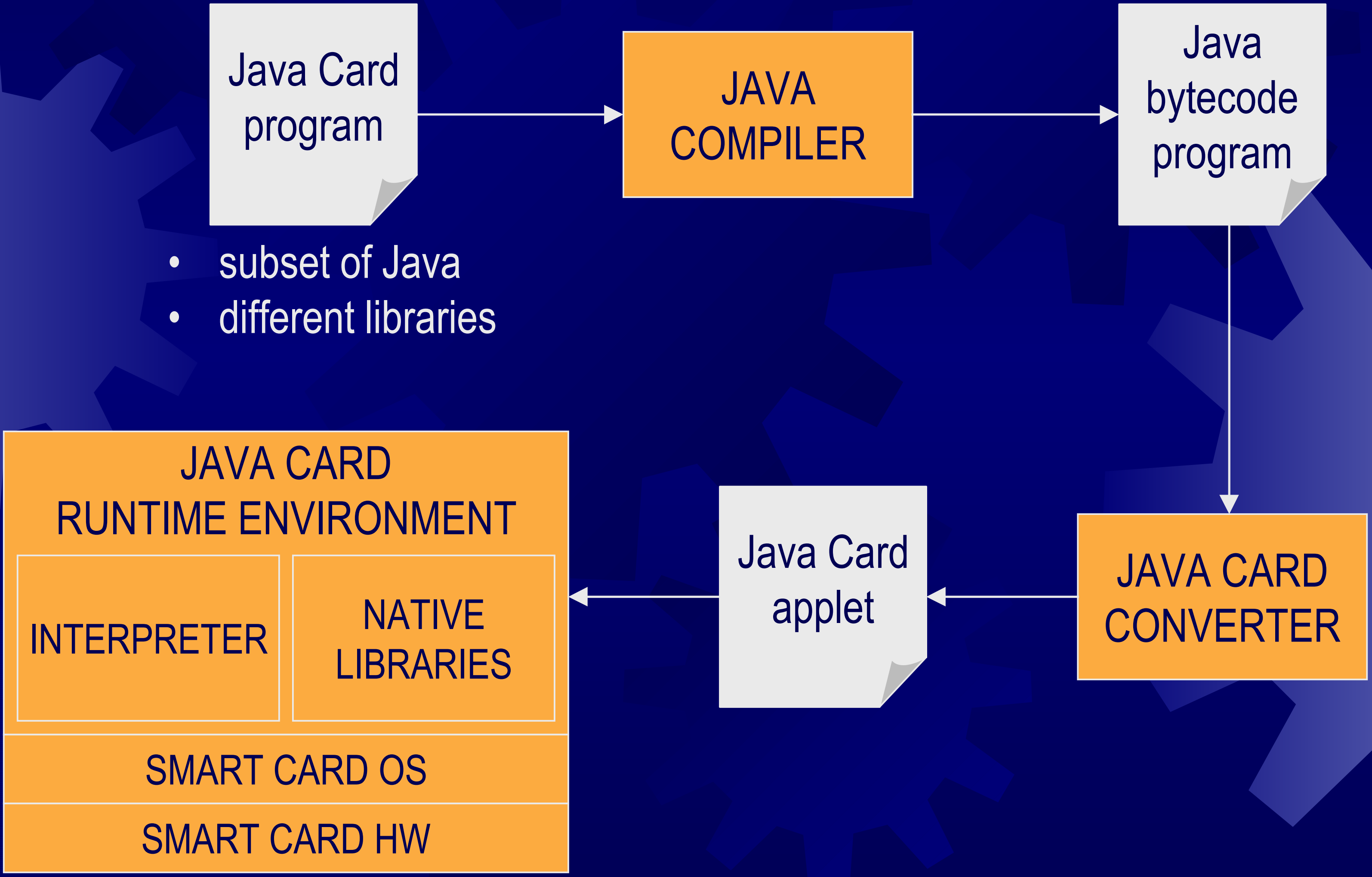
January 2002

What Is Java Card?

A version of Java for smart cards

authentication,
banking,
telephony,
health care,
…

Java Card Technology



- subset of Java
- different libraries

Java Card Libraries

- Standard
  - crypto
  - applet firewall
  - persistent & transient objects
  - atomicity & transactions
  - communication with host terminal
- Industry-specific
  - telephony (GSM)
  - banking
  - …

Why Java for smart cards?

- Many different HW/OS platforms
  - write once, run anywhere
  - strong typing (support for security)
  - multiple vendors
  - post-issuance personalization/update
- Other standards
  - C/MULTOS
  - Windows for Smart Card
    - conjecture: .NET for smart cards?

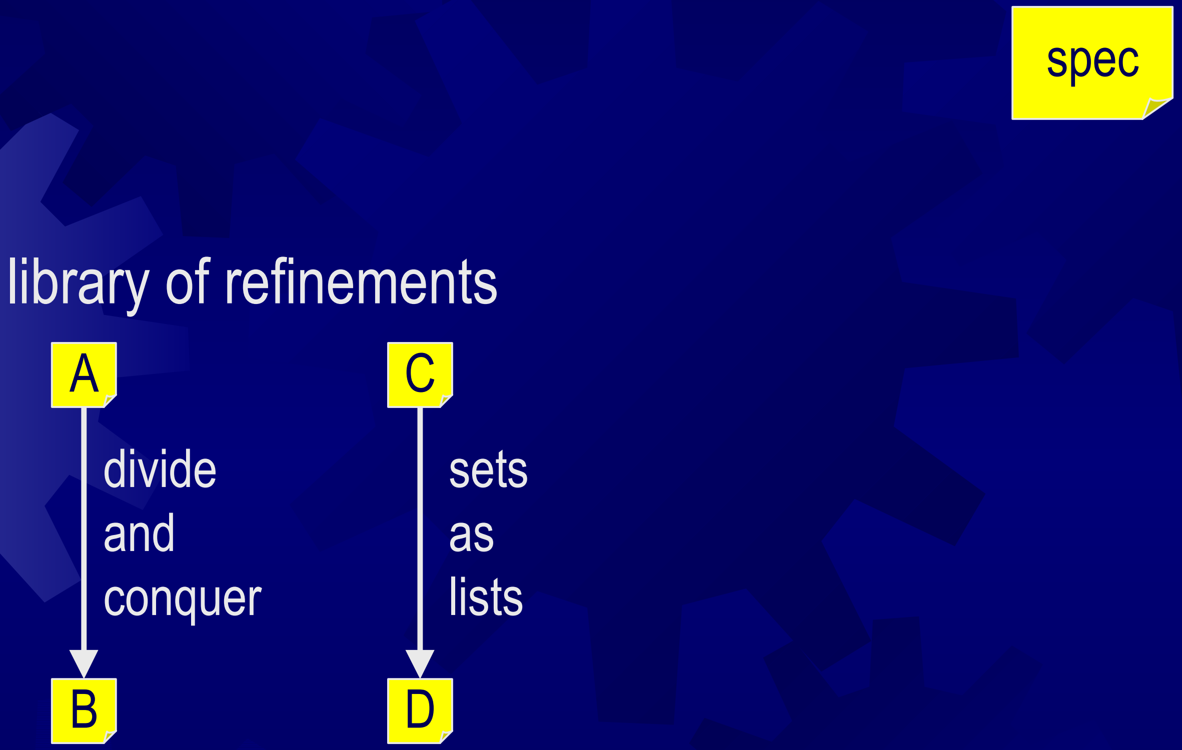
High Assurance

- Critical requirement for smart cards
- Pursued by smart card vendors (Gemplus, Bull, Schlumberger, …)
- Measurable (Common Criteria)
- Focus of Kestrel Institute's research
  - automated synthesis ("specs to code")
  - formal analysis

Kestrel’s Synthesis Systems

- Specware
  - formal specs
  - refinement
  - composition
  - code generation
- Designware
  - libraries of specs and refinements embodying software design knowledge (algorithms, optimizations, …)
  - tactics for automated refinement in Specware
- Planware
  - automatic generator of high-performance, complex resource systems (allocation, transportation schedulers, …)
  - on top of Specware
- MoBIES, HARBINGER, SVA, …

Kestrel’s Synthesis Approach



spec

library of refinements

A

divide
and
conquer

B

C

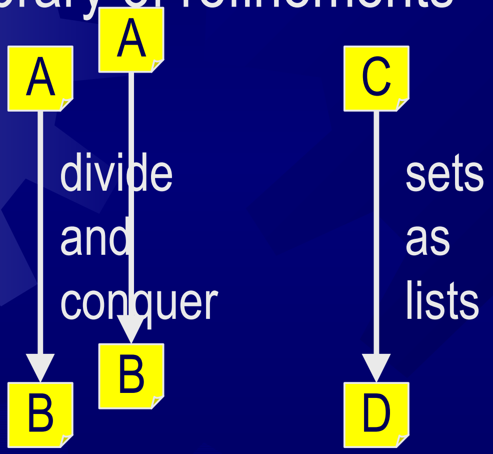
sets
as
lists

D

Kestrel’s Synthesis Approach

spec

library of refinements



A

A

C

divide
and
conquer

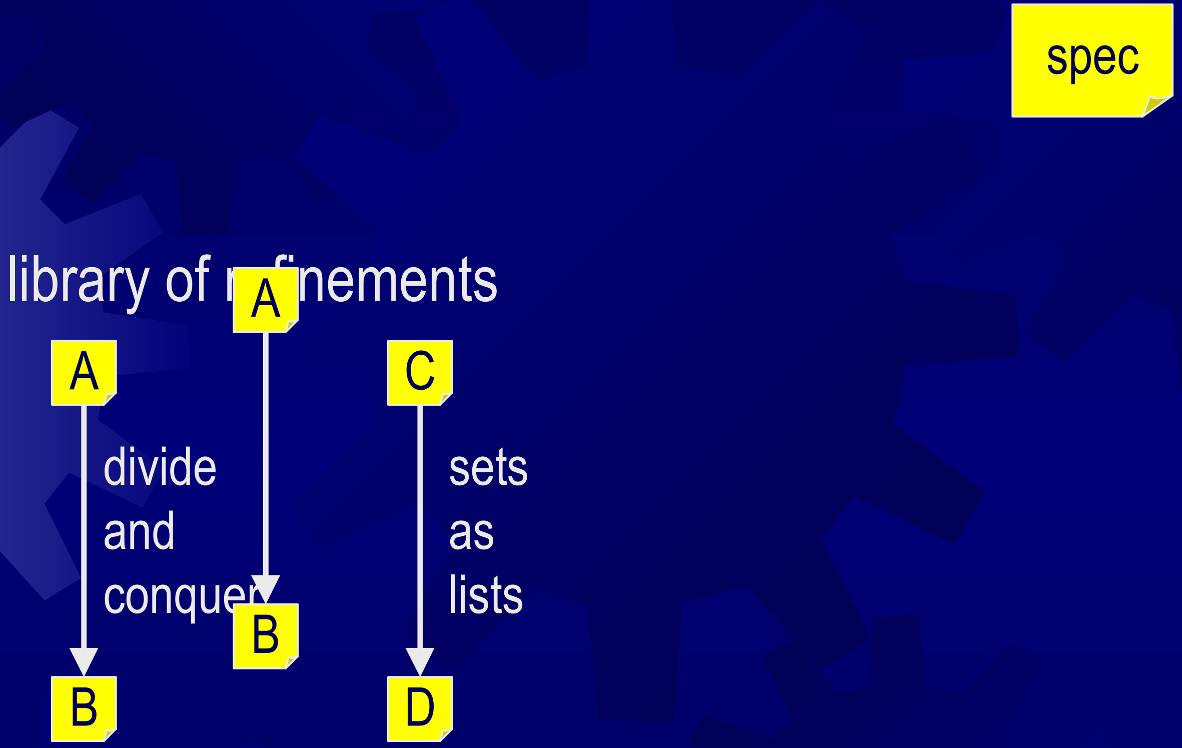
sets
as
lists

B

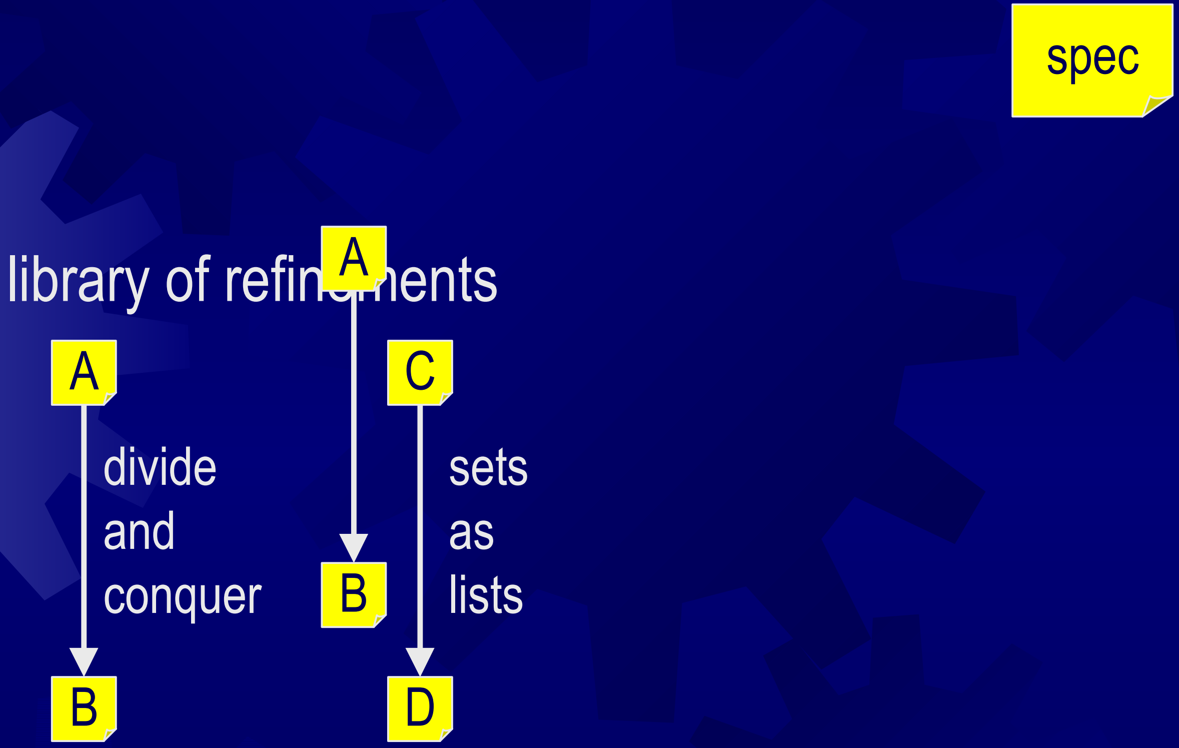
B

D

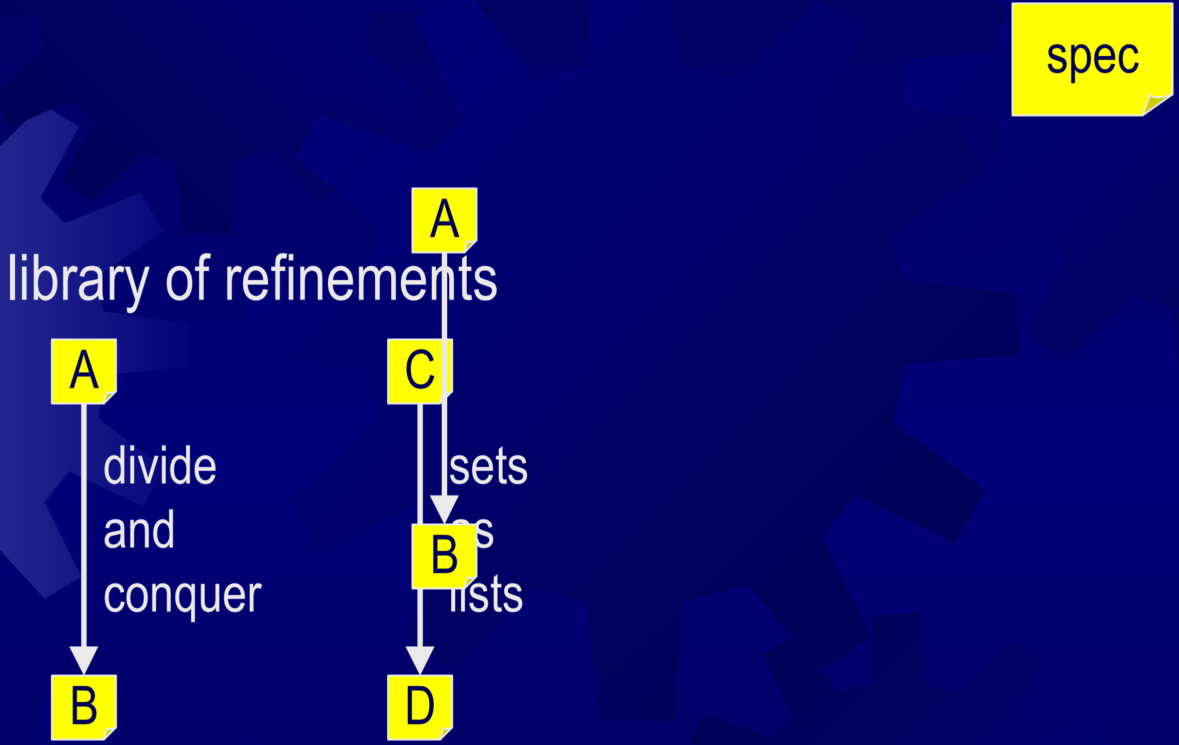
Kestrel's Synthesis Approach



Kestrel’s Synthesis Approach



Kestrel's Synthesis Approach



spec

A

library of refinements

A

divide and conquer

B

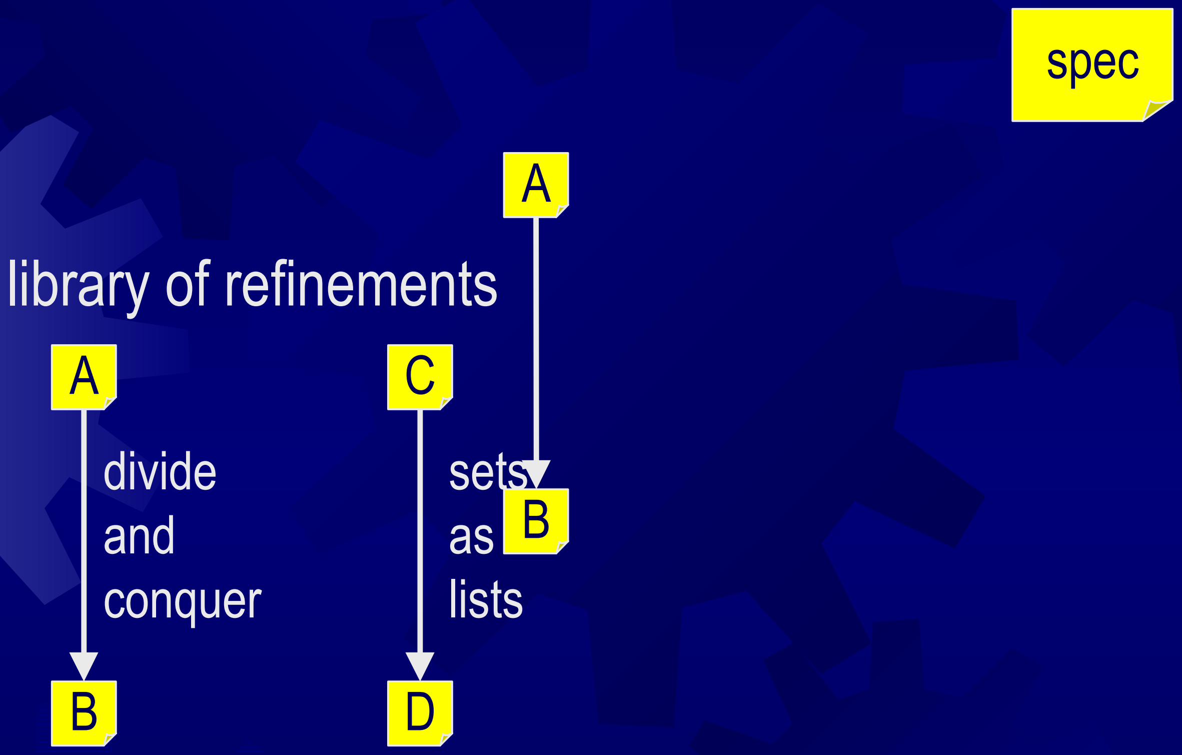
C

sets as lists

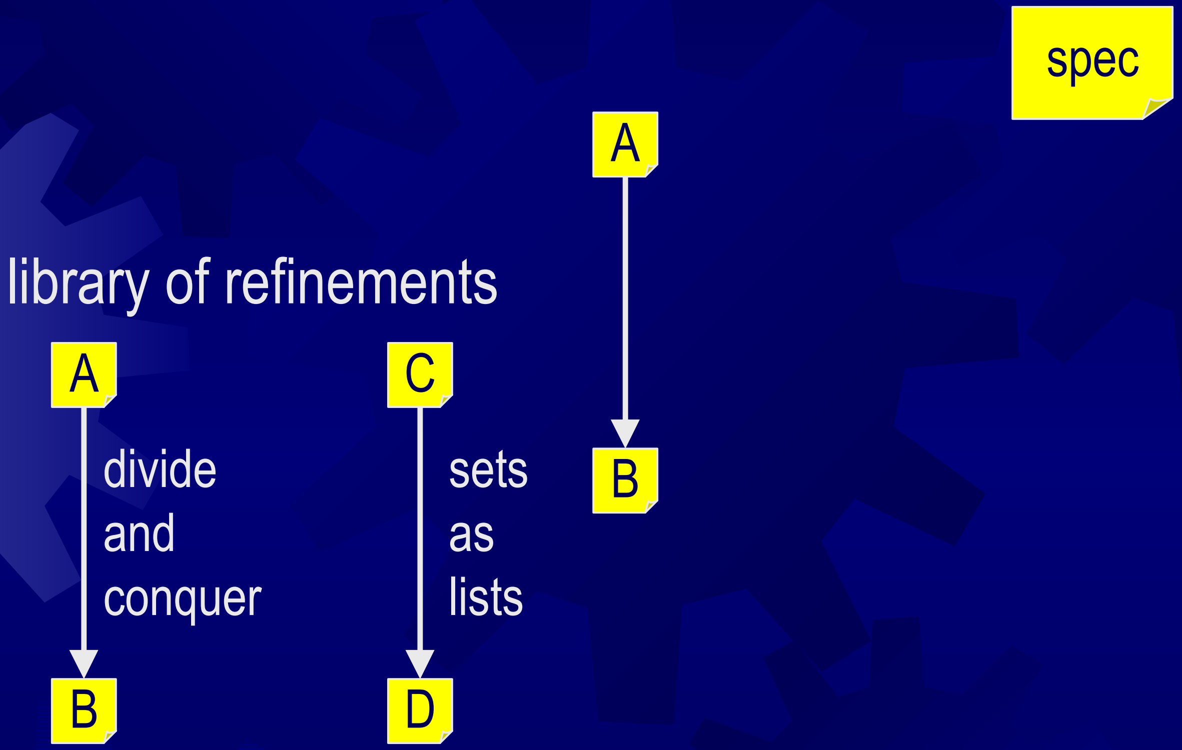
B

D

Kestrel’s Synthesis Approach



Kestrel’s Synthesis Approach



spec

A

B

library of refinements

A

divide and conquer

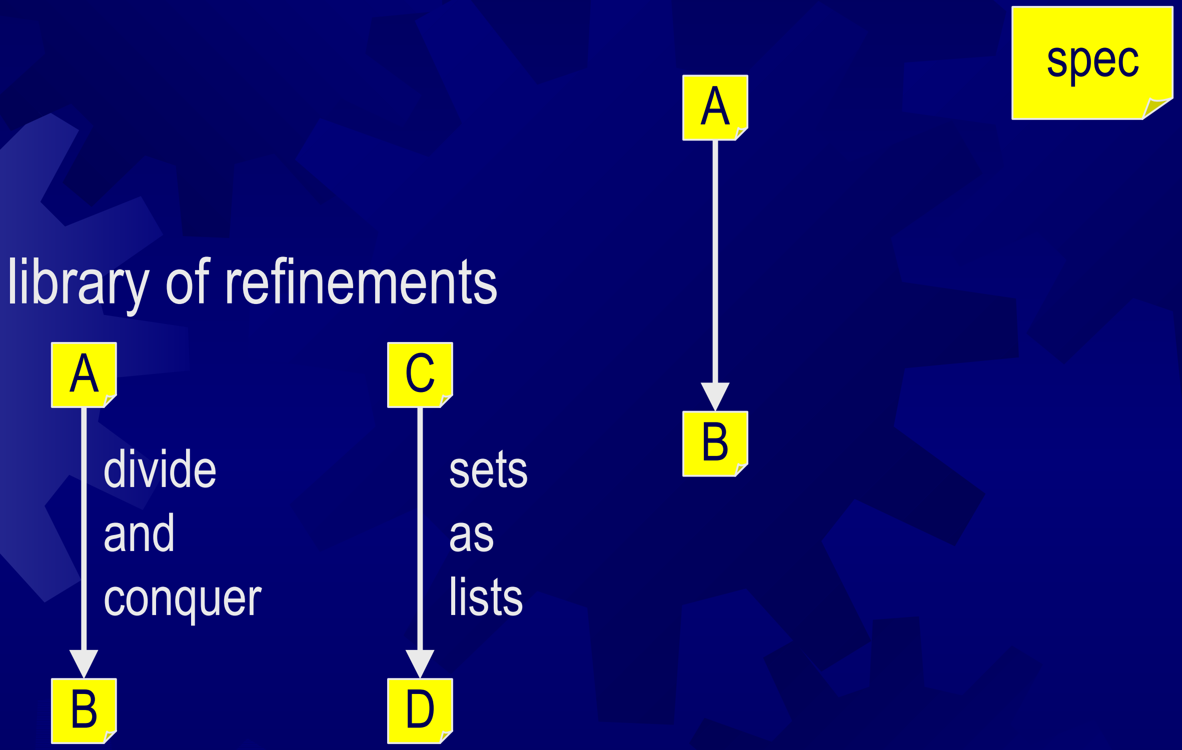
B

C

sets as lists

D

Kestrel’s Synthesis Approach



spec

A

B

library of refinements

A

divide and conquer

B

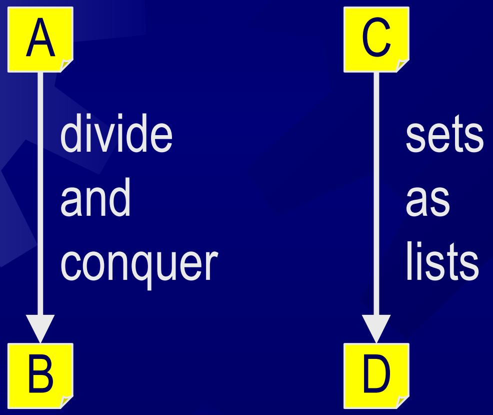
C

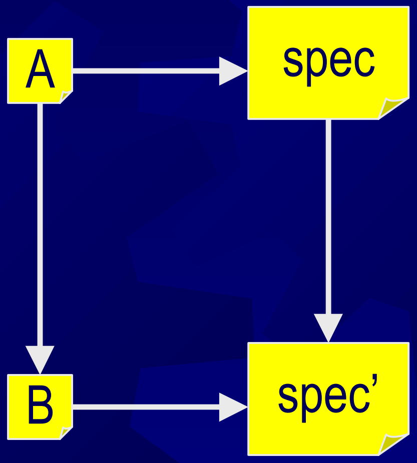
sets as lists

D

Kestrel’s Synthesis Approach

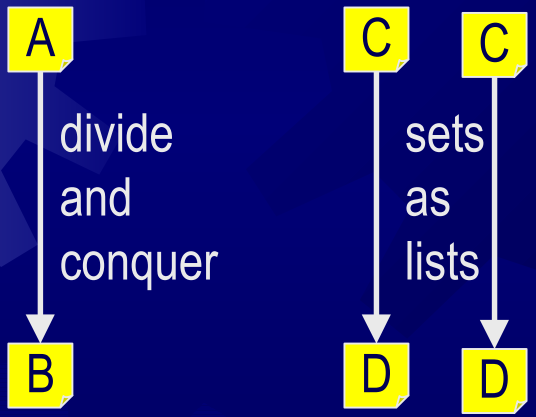
library of refinements

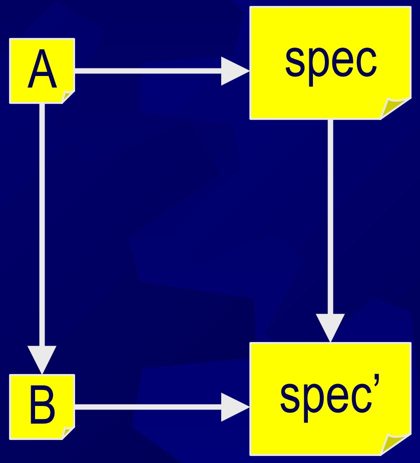




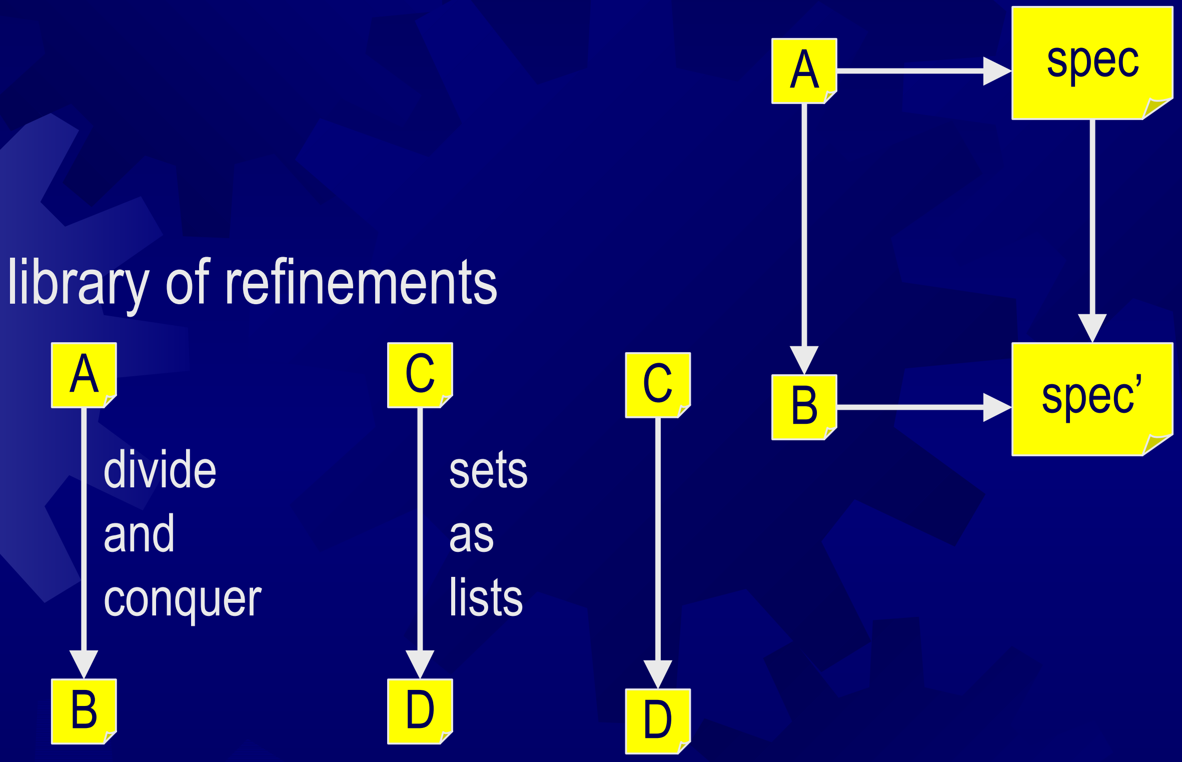
Kestrel’s Synthesis Approach

library of refinements



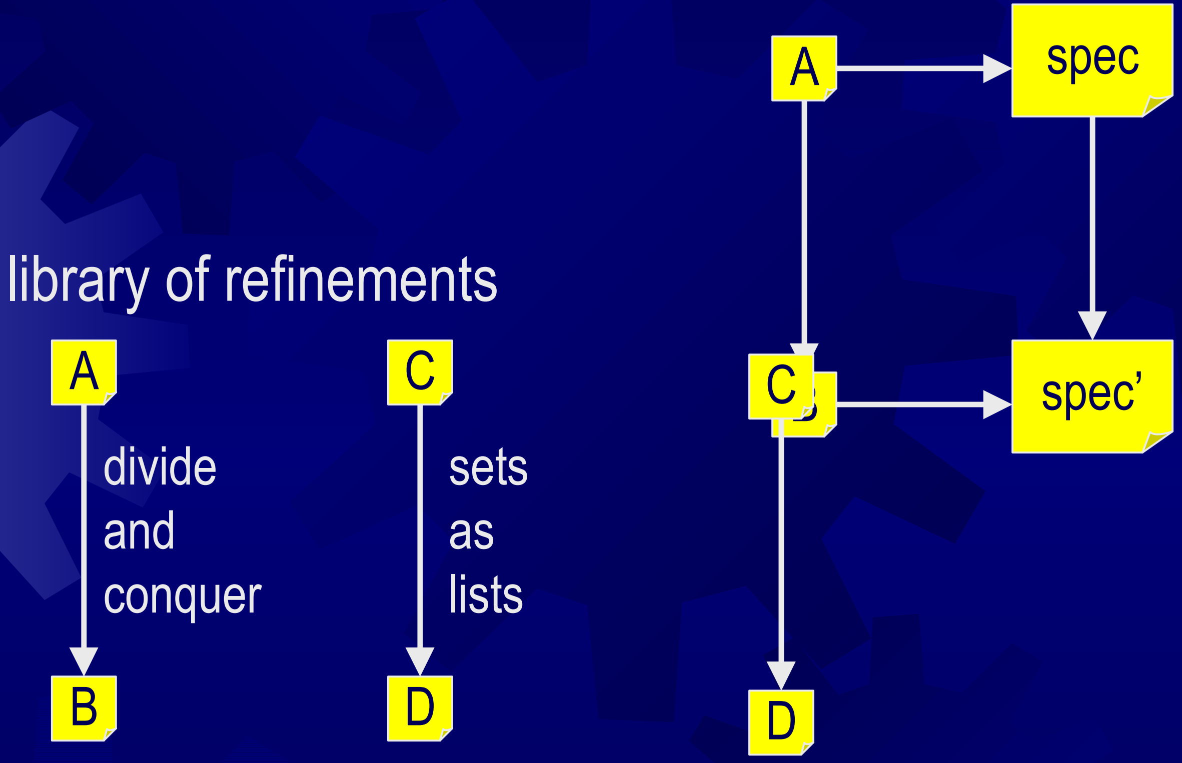


Kestrel’s Synthesis Approach



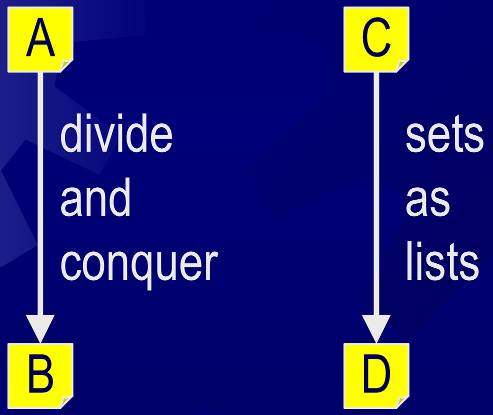
library of refinements

Kestrel’s Synthesis Approach



Kestrel’s Synthesis Approach

library of refinements



A

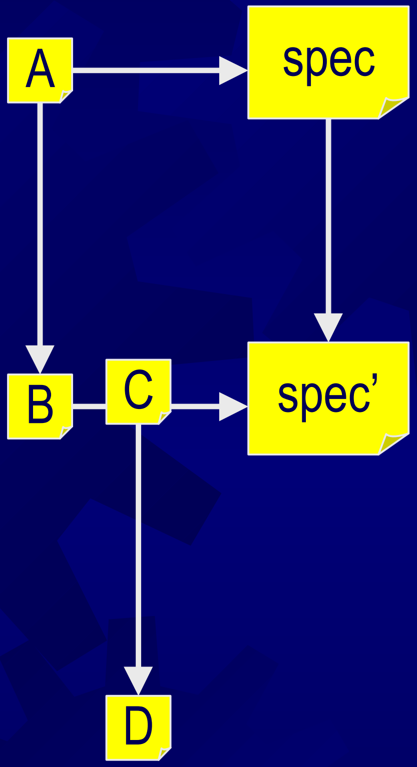
divide
and
conquer

B

C

sets
as
lists

D



A

spec

B

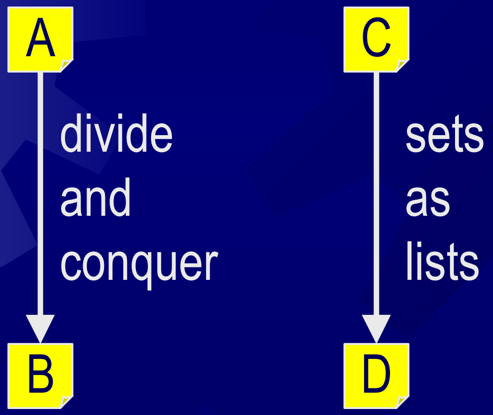
C

spec’

D

Kestrel’s Synthesis Approach

library of refinements



A

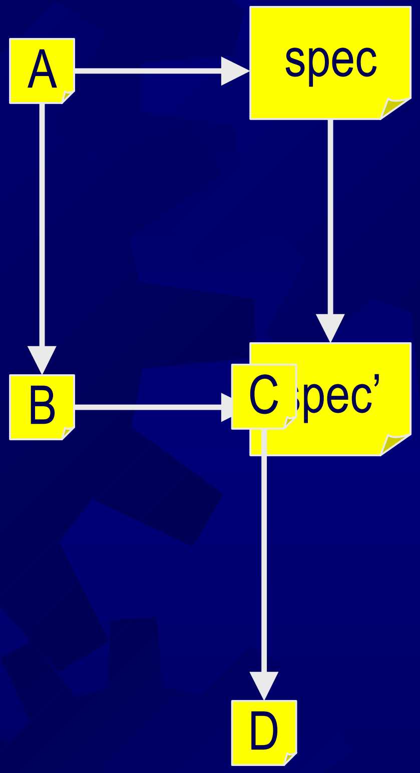
divide
and
conquer

B

C

sets
as
lists

D



A

spec

B

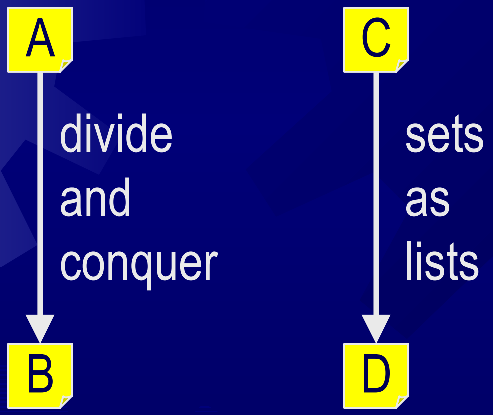
C

spec’

D

Kestrel’s Synthesis Approach

library of refinements



A

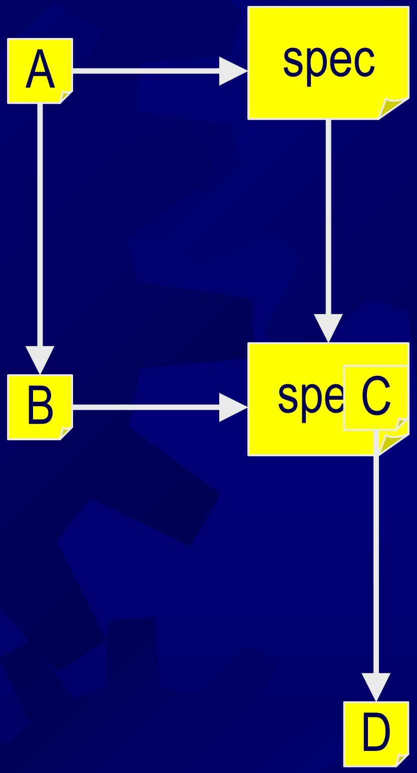
divide and conquer

B

C

sets as lists

D



A

spec

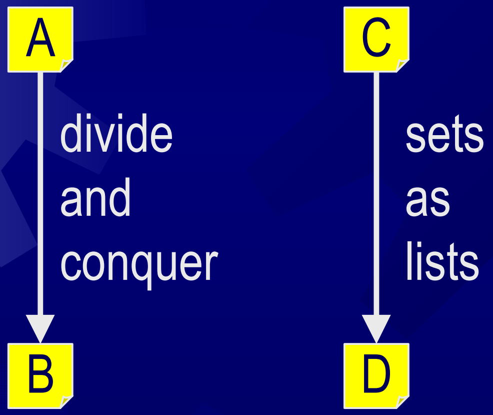
B

spe C

D

Kestrel’s Synthesis Approach

library of refinements



A

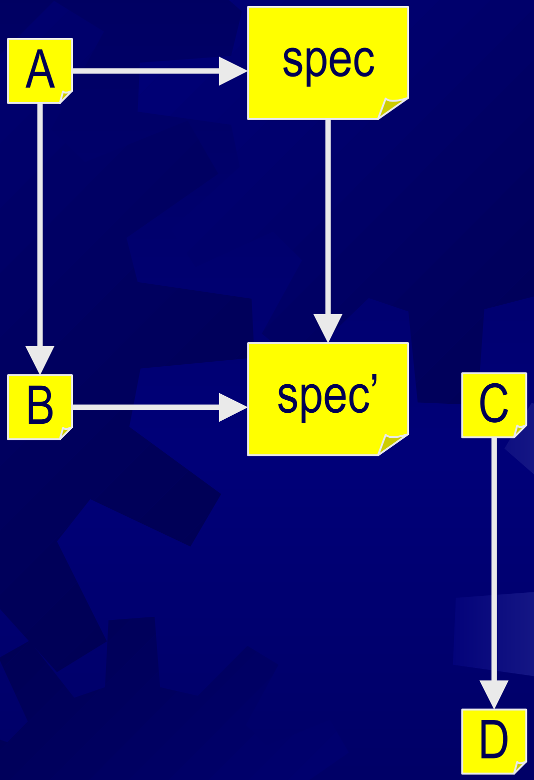
divide
and
conquer

B

C

sets
as
lists

D



A

spec

B

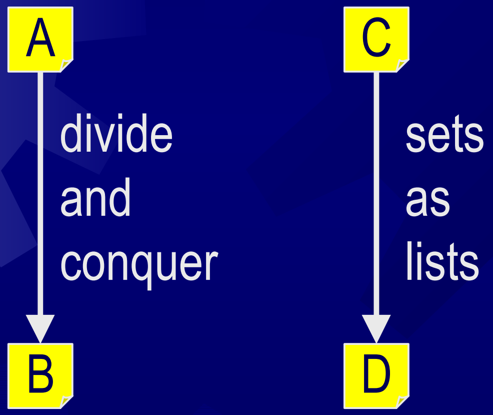
spec’

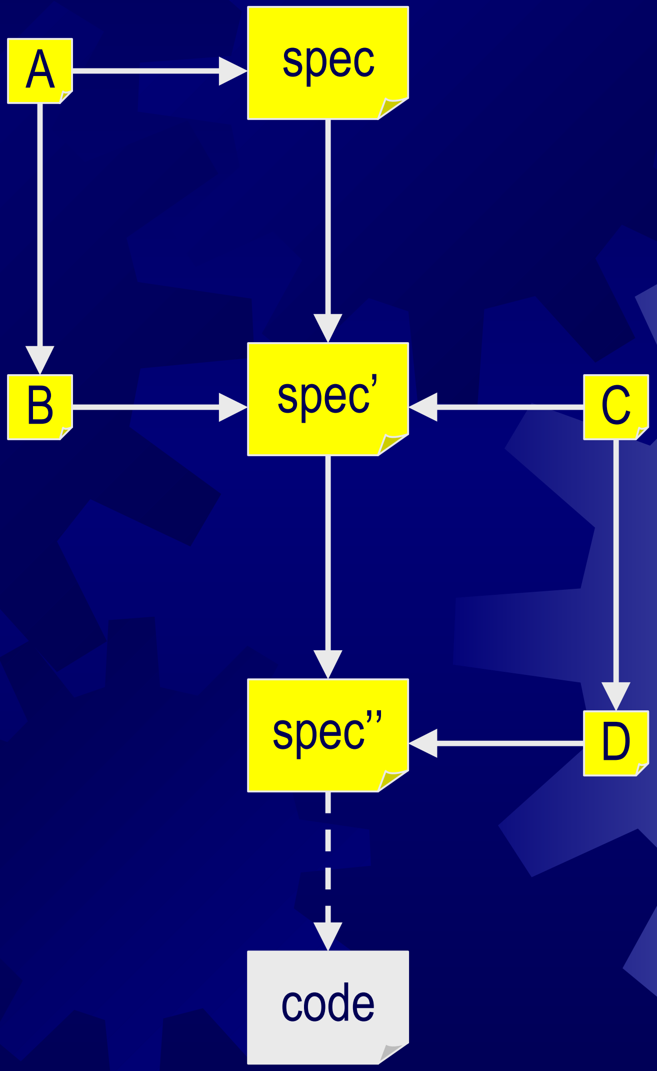
C

D

Kestrel’s Synthesis Approach

library of refinements





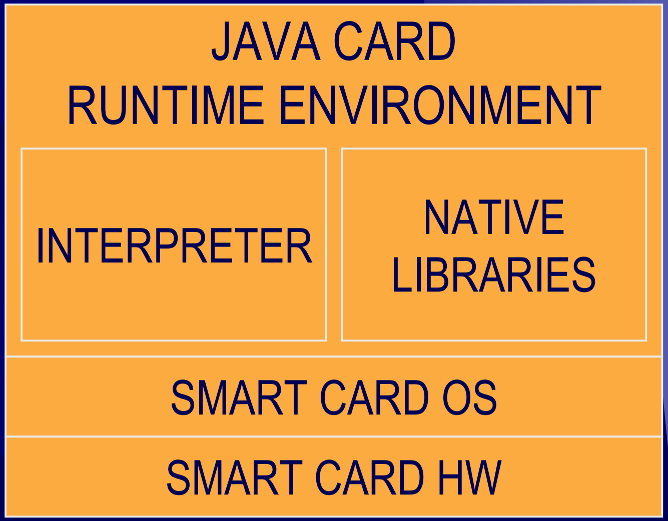
Kestrel’s Past Work on Java

Type safety in the Java Virtual Machine (base for Java security)

- bytecode verification
  - complete verifier in Specware (spec to code)
  - improvements over Sun’s (subroutines, subtyping, …)
  - found bugs in Sun’s spec and verifier
- class loading
  - formal specification
  - type safety theorem
- first
  - formally developed verifier
  - useful spec of class loading

High-Assurance Java Card

- Platform
  - synthesis of
    - Java Card Runtime Environment (JCRE)
    - simulator
    - off-card verifier
    - …
- Applets
  - applet generator



JAVA CARD RUNTIME ENVIRONMENT

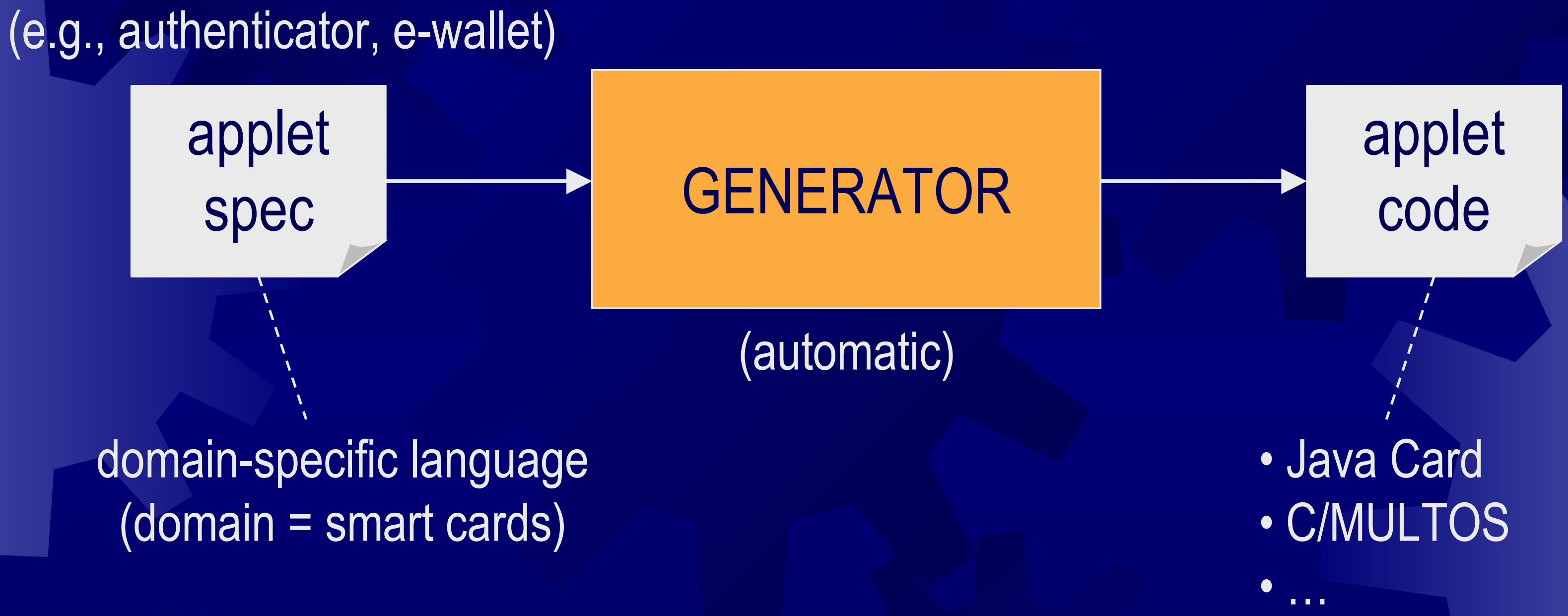
INTERPRETER

NATIVE LIBRARIES

SMART CARD OS

SMART CARD HW

Applet Generator



(e.g., authenticator, e-wallet)

applet spec

GENERATOR

applet code

(automatic)

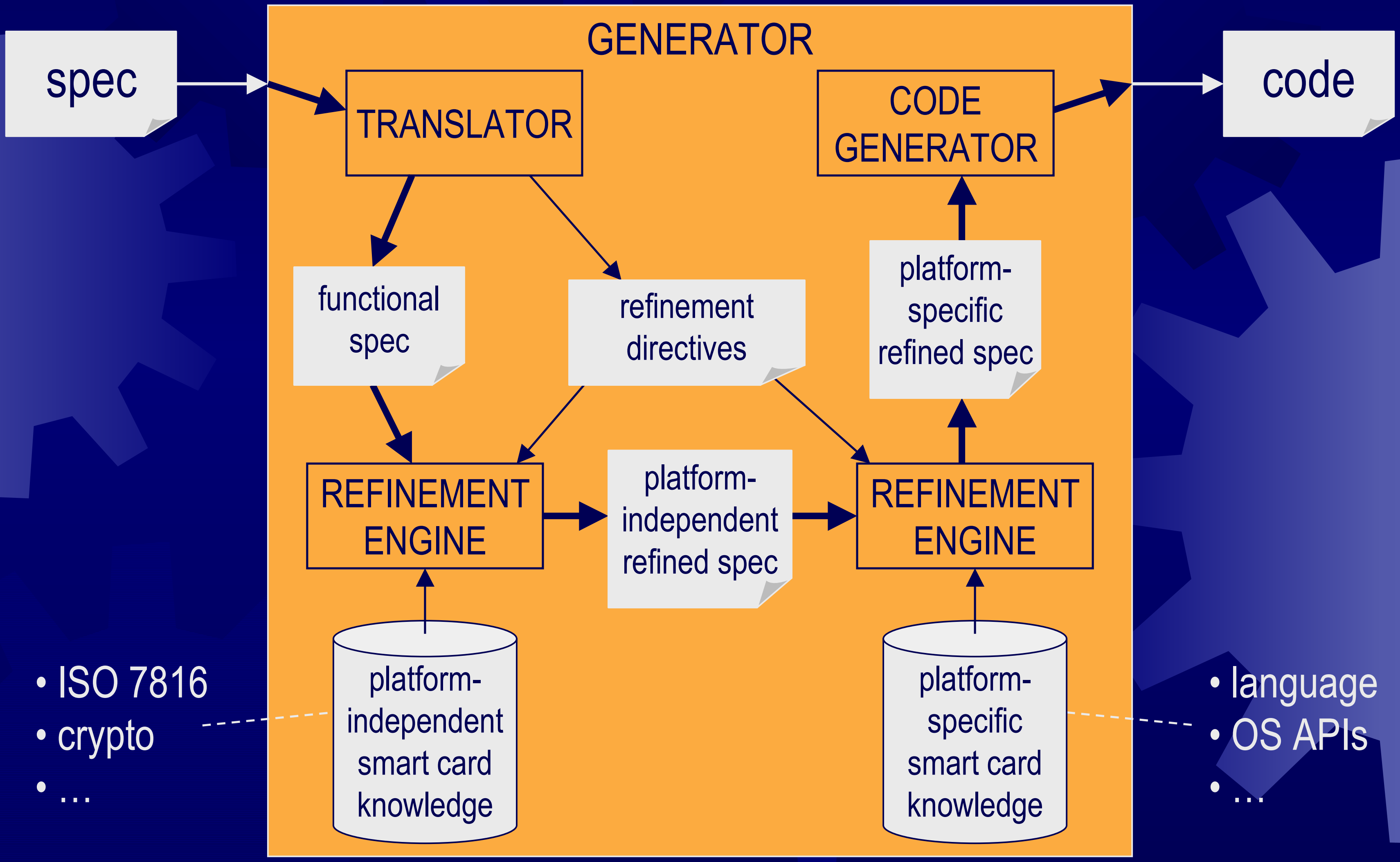
domain-specific language
(domain = smart cards)

- Java Card
- C/MULTOS
- …

for:

- productivity
- high assurance

Specware-Based Approach

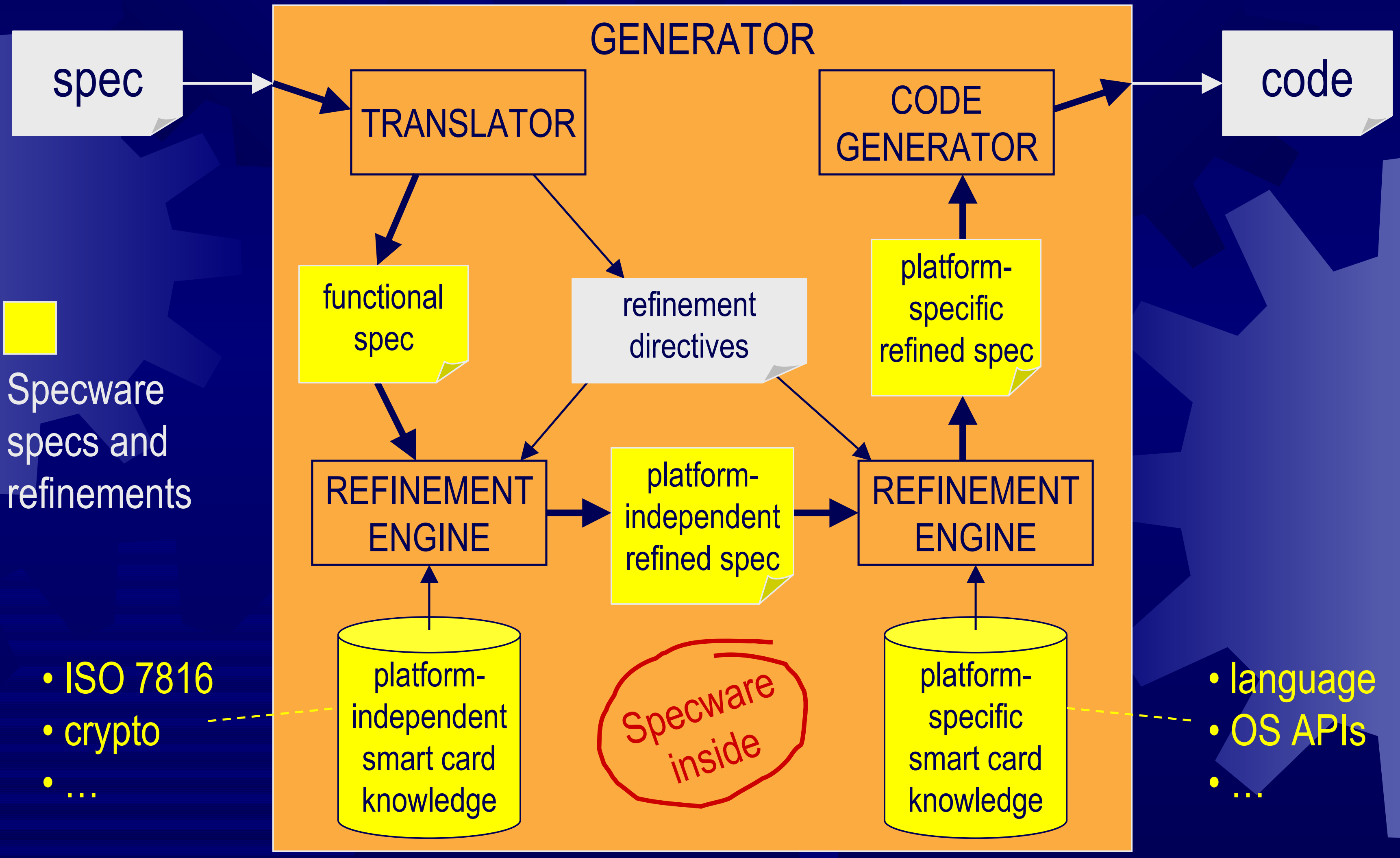


spec
GENERATOR
TRANSLATOR
CODE GENERATOR
code
functional spec
refinement directives
platform-specific refined spec
REFINEMENT ENGINE
platform-independent refined spec
REFINEMENT ENGINE
platform-independent smart card knowledge
platform-specific smart card knowledge

- ISO 7816
- crypto
- …

- language
- OS APIs
- …

Specware-Based Approach



spec

GENERATOR

TRANSLATOR

functional spec

refinement directives

REFINEMENT ENGINE

platform-independent refined spec

REFINEMENT ENGINE

platform-specific refined spec

CODE GENERATOR

code

platform-independent smart card knowledge

platform-specific smart card knowledge

Specware inside

🟨 Specware specs and refinements

- ISO 7816
- crypto
- …

- language
- OS APIs
- …

Example of Applet Derivation

- Functional spec (abstract commands, responses, and states)
- Encoding of commands and responses as APDUs (bytes)
- Refinement of states as bytes
- Introduction of Java Card libraries
- Generation of Java Card code

Advantages of the Approach

- Higher assurance

- synthesis (specs to code)
  - invest in transform correctness
  - get repeated benefit by re-use
  - mathematical foundations

- analysis (write & verify)
  - bad combinatorics
  - little or no re-use
  - hard to infer all properties

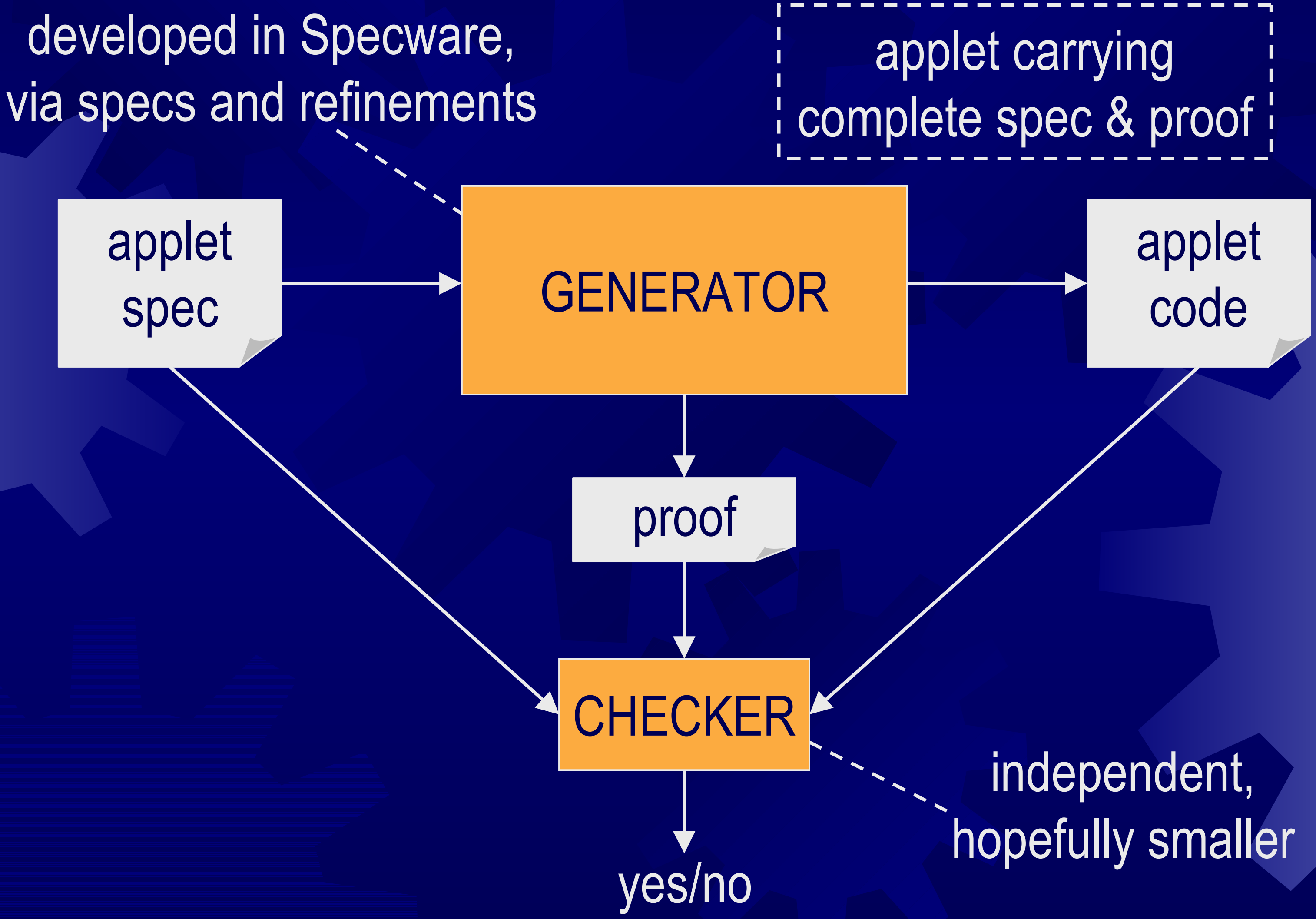
Why Not Develop Library Components to Build Applets?

- Optimization
  - synthesis produces code optimized for
    - size
    - speed
- Large variability in applet functionality
  - hard to predict all needed components
- Security properties
  - synthesis produces proof for whole system

Advantages of the Approach (cont’d)

- Easier to evolve the generator
  - evolve internal knowledge, e.g.
    - add inter-applet communication
    - add new platform (C/MULTOS)
  - evolve individual components, e.g.
    - more platform-specific optimizations
    - smaller footprint of generated code
- Previously successful in Planware
- Independent certification

Independent Certification



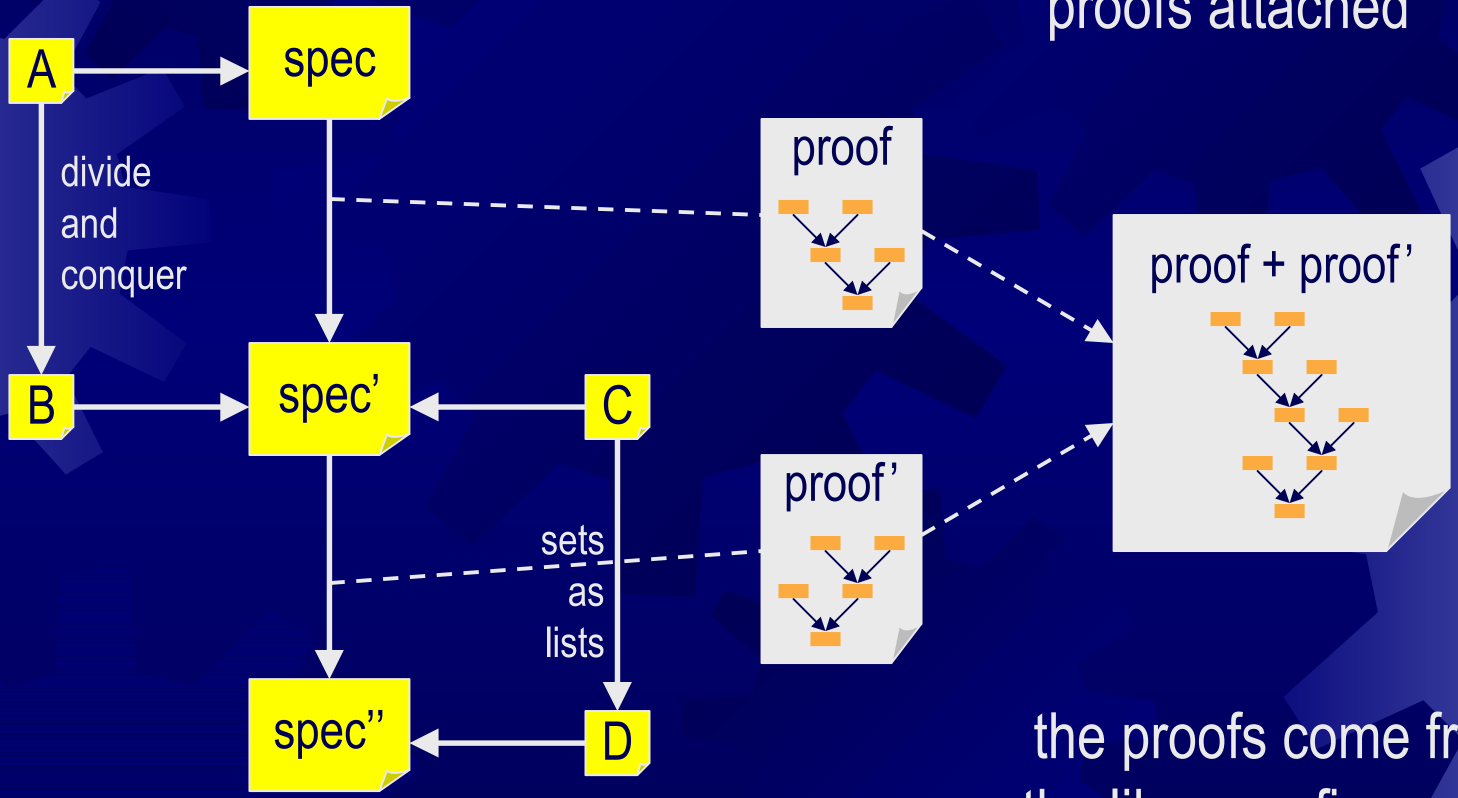
developed in Specware,
via specs and refinements

applet carrying
complete spec & proof

independent,
hopefully smaller

How Do We Build the Proof?

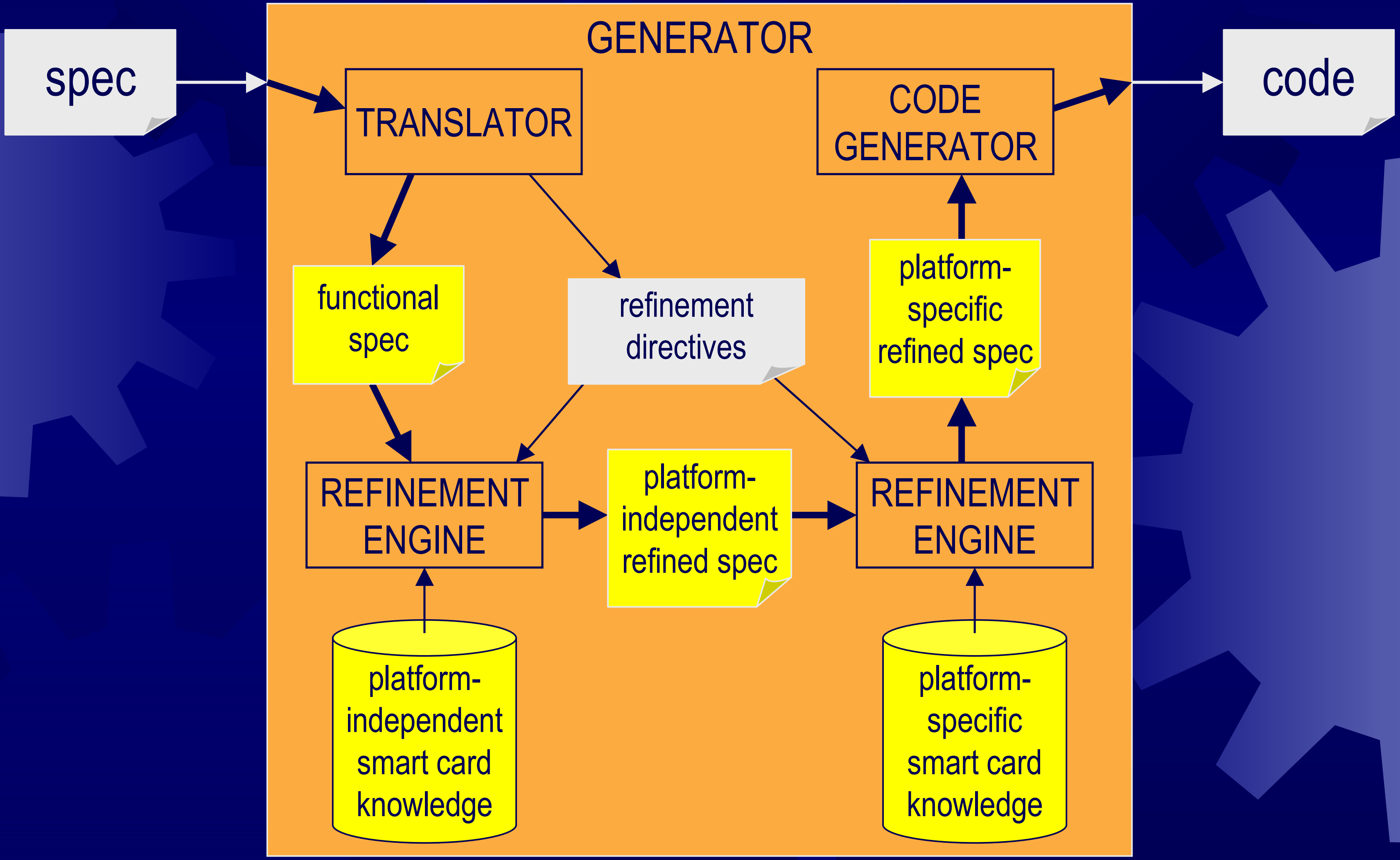
refinements have proofs attached



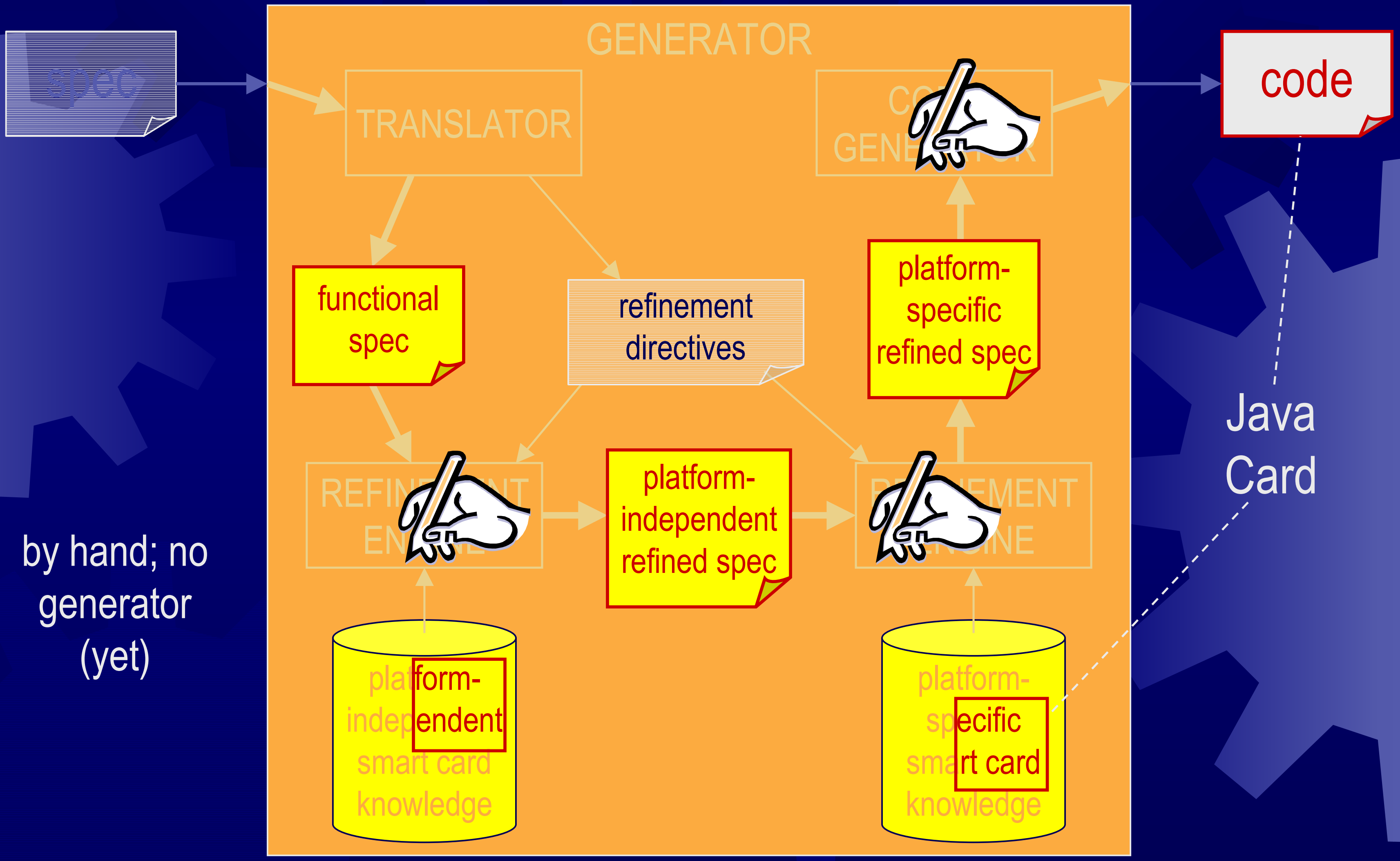
the proofs come from the library refinements (i.e., re-use of proofs)

scalability ⇐

Initial effort:


spec
GENERATOR
TRANSLATOR
CODE GENERATOR
code
functional spec
refinement directives
platform-specific refined spec
REFINEMENT ENGINE
platform-independent refined spec
REFINEMENT ENGINE
platform-independent smart card knowledge
platform-specific smart card knowledge

Initial effort : Complete Spec-to-code CAC Applet



spec

GENERATOR

TRANSLATOR

functional spec

refinement directives

REFINEMENT ENGINE

platform-independent refined spec

REFINEMENT ENGINE

platform-specific refined spec

CODE GENERATOR

code

platform-independent smart card knowledge

platform-specific smart card knowledge

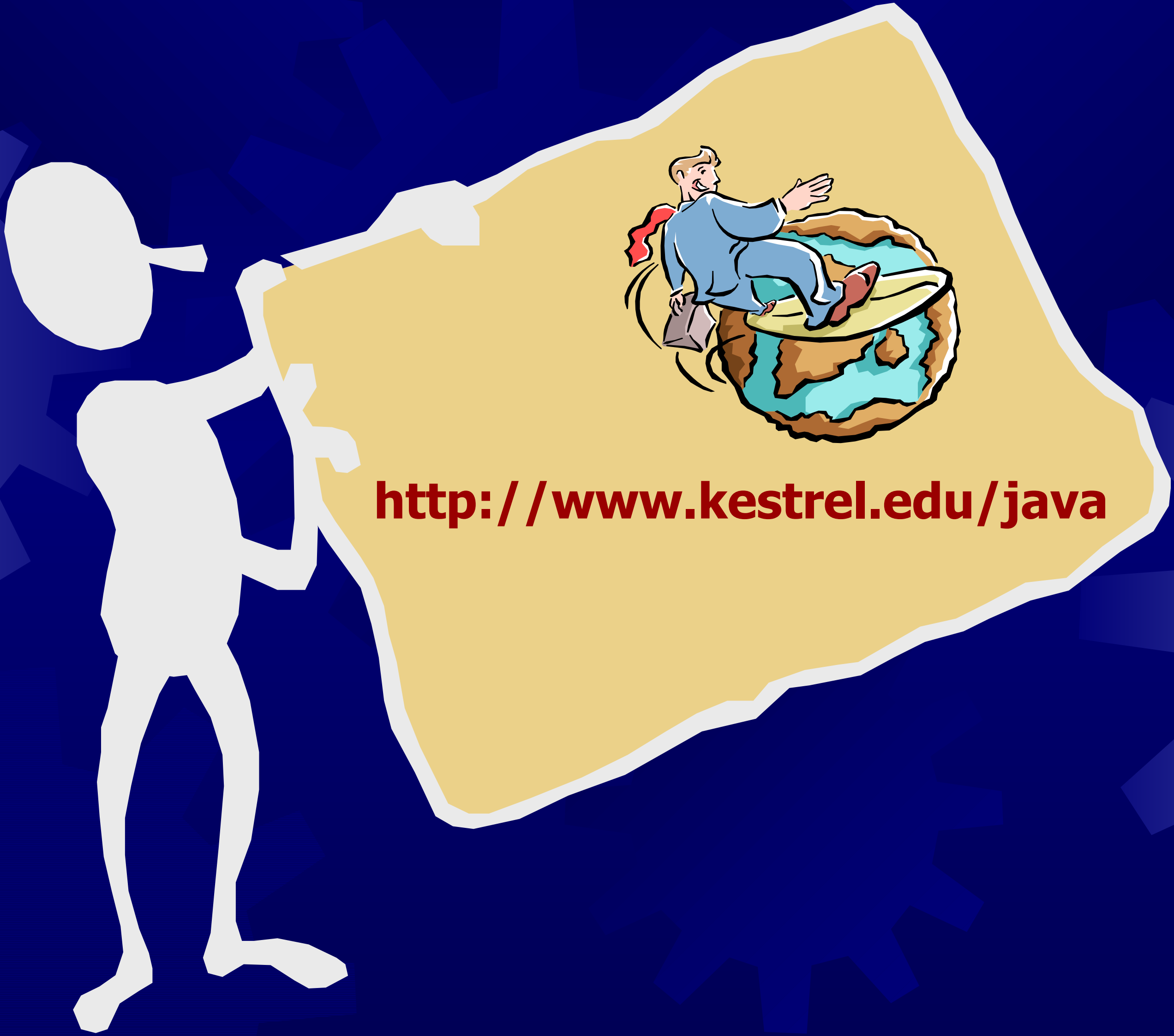
Java Card

by hand; no generator (yet)

Purpose of This Initial Effort

- Determine initial fundamental specs and refinements needed
- Elaborate patterns/structure of such specs and refinement construction
- Develop applet design knowledge (e.g., theories and refinements for ISO 7816)
- Build 1st version of generator based on the above

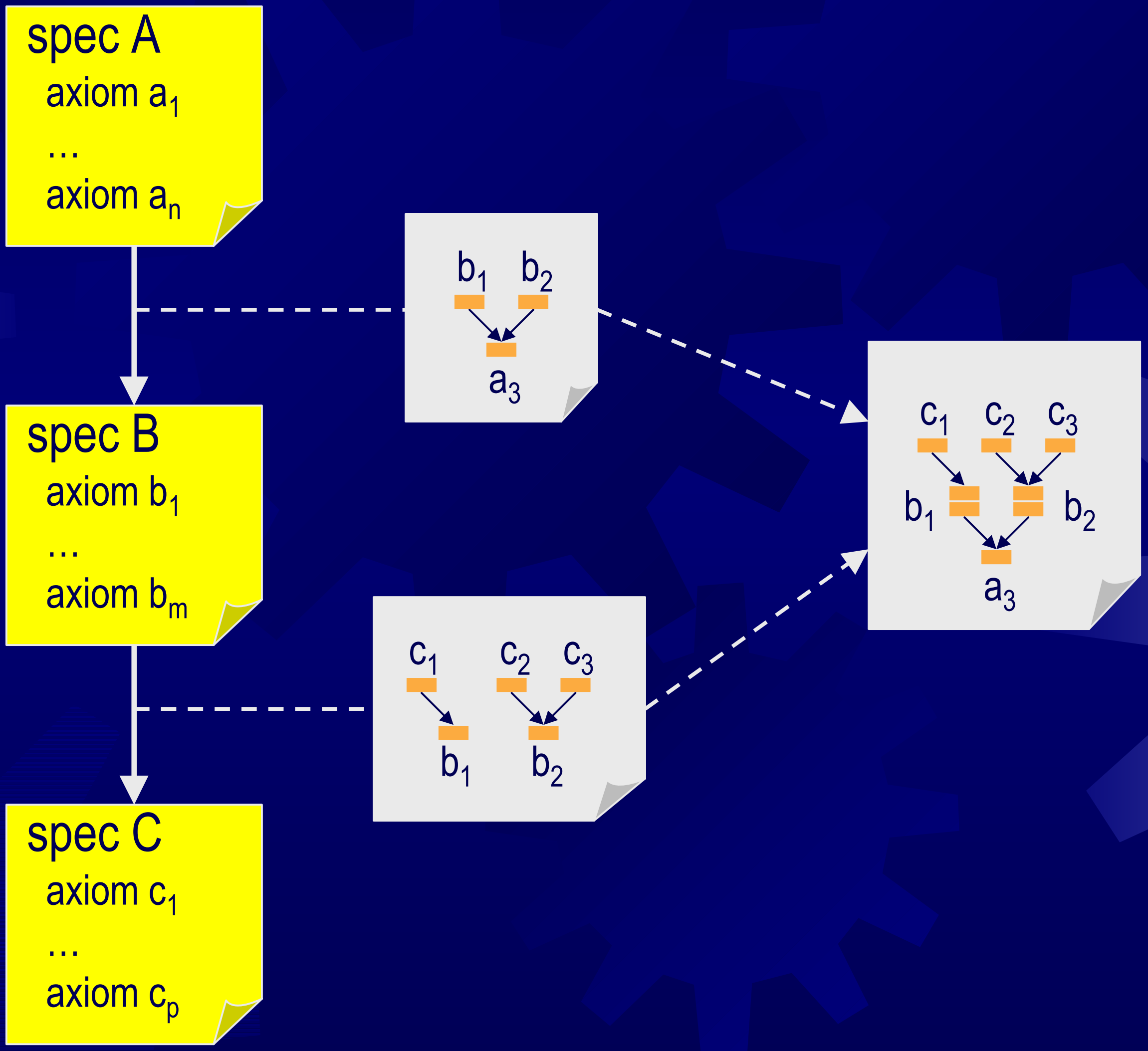
For More Information



http://www.kestrel.edu/java

Backup Slides

Proof Composition: Example



spec A
axiom a_1
...
axiom a_n

spec B
axiom b_1
...
axiom b_m

spec C
axiom c_1
...
axiom c_p

Synthesis of JCRE & Tools

- Use of Specware
- Spec of JCRE
  - refinement to simulator (runs on PC)
  - refinement to smart card HW/OS
- Off-card verifier
  - leverage of our JVM bytecode verifier
  - approaches to put it on card (security ↑)

Results to Date

- Working CAC applet
- Ready to build 1st version of generator
- Integration with other Kestrel work for
  - stateful specs and refinements
  - generation of (maintainable) Java code
- Integration of Specware-generated code with external libraries (APIs)