Synthesis of Provably Correct Java Card Applets and Platform

Alessandro Coglio

Kestrel Institute

HCSS Conference

April 15th, 2004

what is synthesis?

- higher level
- simpler, smaller
- easier to get right
- more re-usable
- easier to change

spec

synthesis

code

- lower level
- larger, more complex
- may include optimizations
- large/spread change for small change in spec

– automation +

general-purpose synthesis

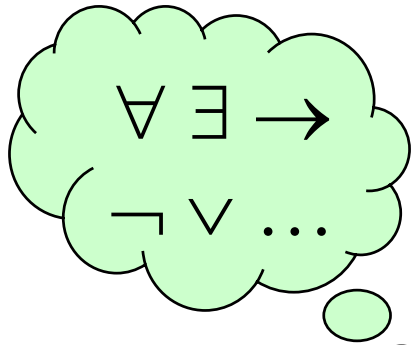
domain-specific synthesis

(as opposed to just writing the code)

code satisfies spec

provable correctness

(high assurance)



- higher level
- simpler, smaller
- easier to get right
- more re-usable
- easier to change

spec

formal synthesis

code

- lower level
- larger, more complex
- may include optimizations
- large/spread change for small change in spec

– automation +

general-purpose synthesis

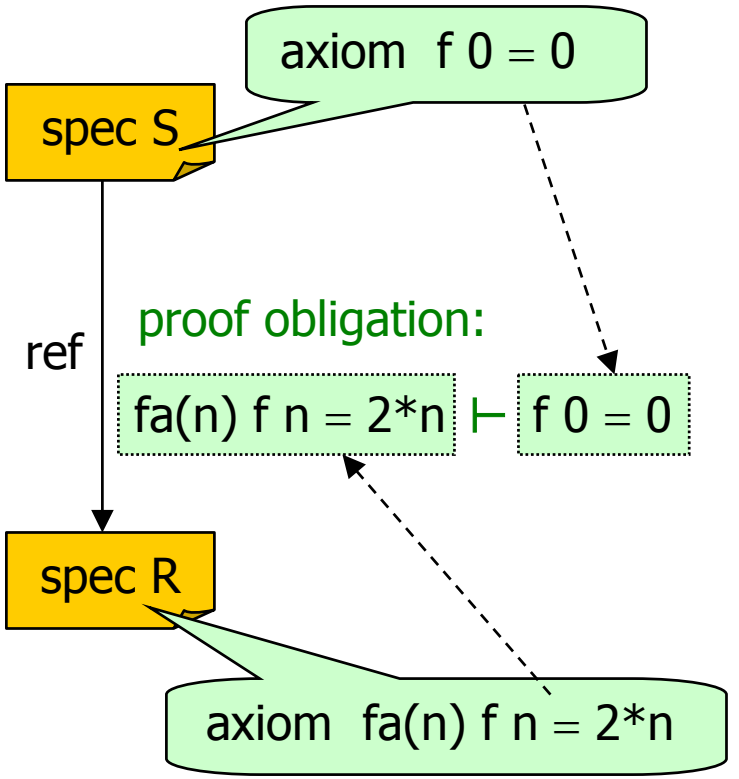
domain-specific synthesis

(as opposed to just writing the code)

formal synthesis = Kestrel's research focus

our flagship  tool:

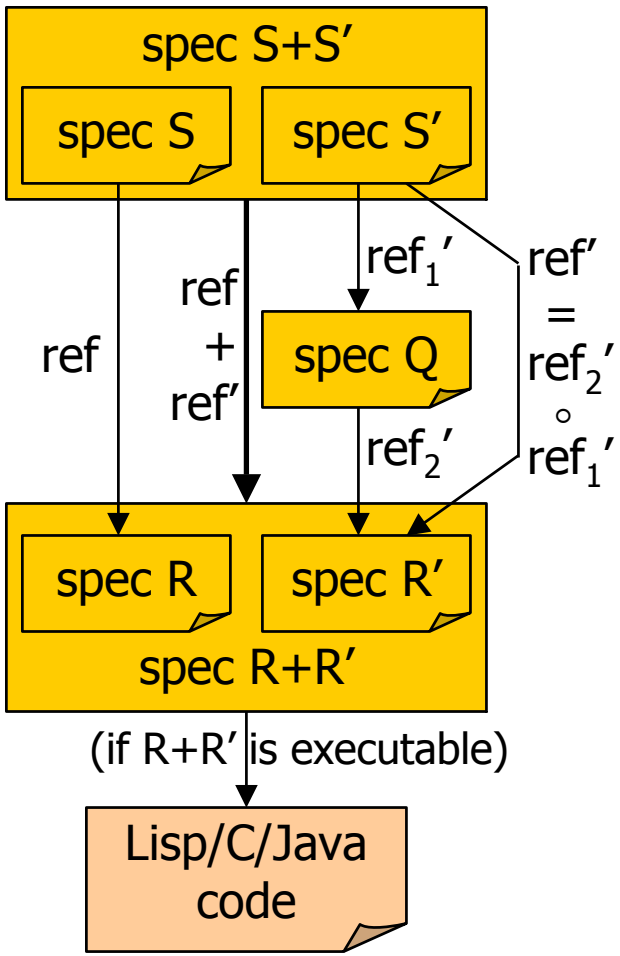
SPECWARE™



- general-purpose synthesis
- some automation 
  but user makes "creative" steps 
- spec = higher-order logic theory λ
- refinement = theory interpretation

formal synthesis = Kestrel's research focus

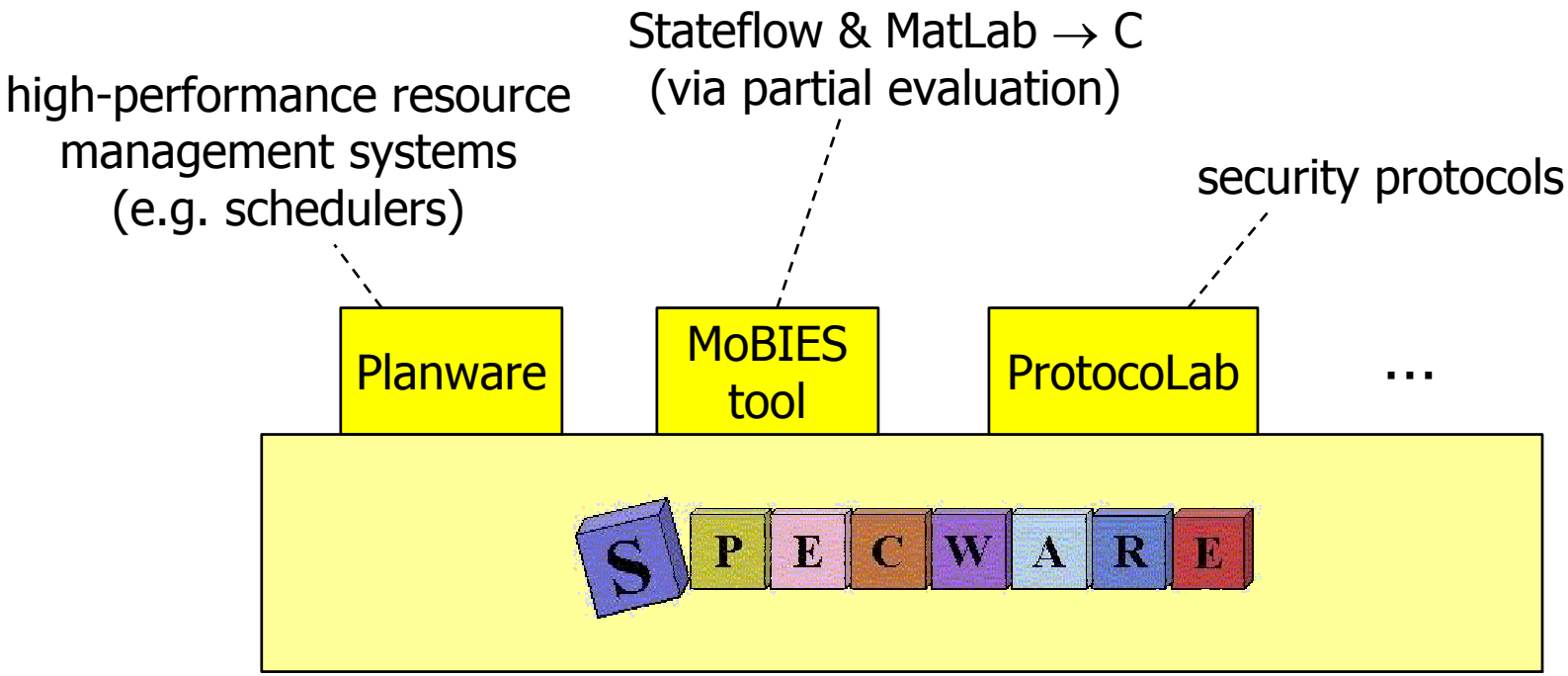
our flagship tool: SPECWARE™



- general-purpose synthesis
- some automation but user makes "creative" steps
- spec = higher-order logic theory λ
- refinement = theory interpretation
- operations to compose specs & refinements + ∘ …
- interfaces to theorem provers to discharge proof obligations ⊢
- Lisp/C/Java code generated from executable specs

formal synthesis = Kestrel's research focus

our domain-specific generators:

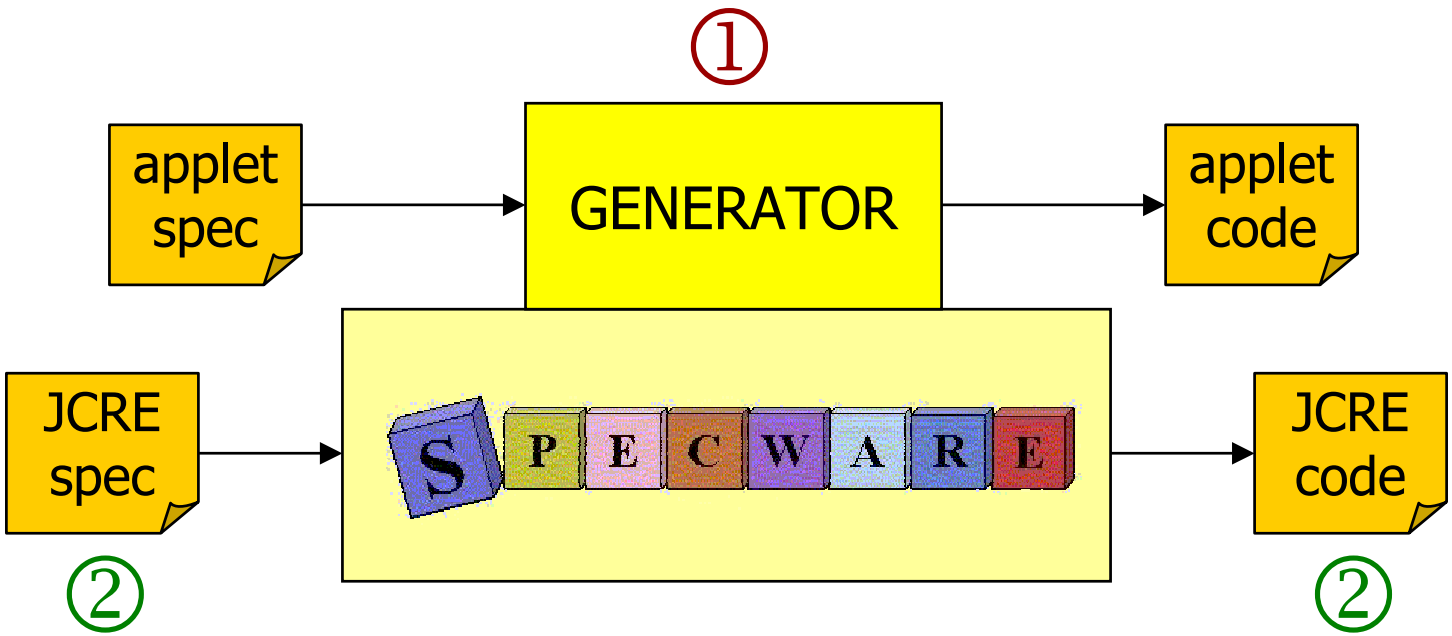


(built on top of Specware)

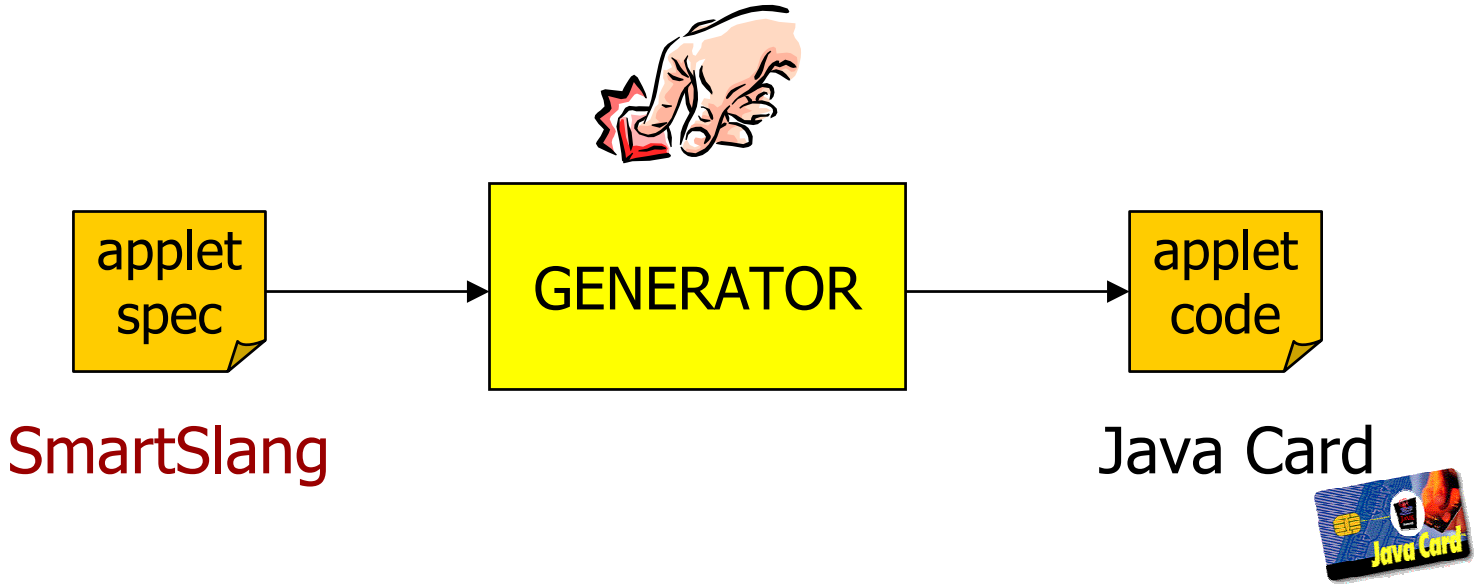
ongoing  project at Kestrel 

① Java Card applet generator

② synthesis of Java Card Runtime Environment



applet generator



- high assurance
- productivity

SmartSlang

domain-specific language

domain = smart cards
i.e. constructs specialized to smart card applications

under design

basic version done,
many additional constructs planned

precise semantics

in terms of state machines

$$applet : Command \times State \rightarrow Response \times State$$

but no formal background necessary

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;
```

1000000

expressive type system
e.g. integer ranges
(vs. byte/short/int)

- capture semantics
- automatic mapping to Java Card types
  e.g. Balance ↦ ~~short~~ int
  Amount ↦ byte

(short, short)

pervasive change in Java Card code

```
state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }
```

explicit state components

```
command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

explicit symbolic commands with high-level parameters (vs. bytes in APDUs)

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

- familiar to developers
- "superset of subset of Java"

simple Java-like expressions & statements

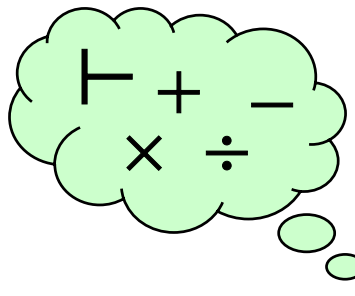
SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

explicit responses

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```



all type safety checked statically (conservatively)

- no runtime errors
- fancy type checker
- user annotations?

type-safe assignments

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

declarative APDU encoding
(vs. explicit decoding/dispatch)

Java Card code for
decoding/dispatching
automatically generated

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

CLA

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

INS

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

P1 & P2

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

data field (Lc = length)

Java Card code
for error handling
(e.g. if amount > 100)
automatically generated

SmartSlang example (e-wallet)

```
type Balance = Int(0,MAX_BALANCE);
type Amount = Int(1,MAX_AMOUNT);

const MAX_BALANCE = 10000;
const MAX_AMOUNT = 100;

state { Balance balance }

command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};

const EXCEEDED_BALANCE = 0x6A84;

...
```

Le

e-wallet example

$$\frac{\text{size(code)}}{\text{size(spec)}} = \sim 3\text{–}4$$

~ 40 lines

SmartSlang spec

GENERATOR

~ 150 lines

Java Card code

(actual files; font size 2)

e-wallet Java Card code

```
...
void credit (APDU apdu) {
  byte[] buffer = apdu.getBuffer();
  if ((buffer[ISO7816.OFFSET_P1] != 0) ||
      (buffer[ISO7816.OFFSET_P2] != 0))
    ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
  inDataLength =
    nonNegativeByte(buffer[ISO7816.OFFSET_LC]);
  if (inDataLength != 1)
    ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
  receiveIncomingData(apdu);
  short amount = nonNegativeByte(inData[0]);
  if ((amount < 1) || (amount > 100))
    ISOException.throwIt(ISO7816.SW_WRONG_DATA);
  if ((short)(balance + amount) <=  MAX_BALANCE)
    balance = (short)(balance + amount);
  else
    ISOException.throwIt((short)EXCEEDED_BALANCE);
}
...
```

e-wallet Java Card code

```
...
void credit (APDU apdu) {
  byte[] buffer = apdu.getBuffer();
  if ((buffer[ISO7816.OFFSET_P1] != 0) ||
      (buffer[ISO7816.OFFSET_P2] != 0))
    ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
  inDataLength =
    nonNegativeByte(buffer[ISO7816.OFFSET_LC]);
  if (inDataLength != 1)
    ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
  receiveIncomingData(apdu);
  short amount = nonNegativeByte(inData[0]);
  if ((amount < 1) || (amount > 100))
    ISOException.throwIt(ISO7816.SW_WRONG_DATA);
  if ((short)(balance + amount) <=  MAX_BALANCE)
    balance = (short)(balance + amount);
  else
    ISOException.throwIt((short)EXCEEDED_BALANCE);
}
...
```

interesting computation

e-wallet Java Card code

APDU checking & decoding

```
...
void credit (APDU apdu) {
  byte[] buffer = apdu.getBuffer();
  if ((buffer[ISO7816.OFFSET_P1] != 0) ||
      (buffer[ISO7816.OFFSET_P2] != 0))
    ISOException.throwIt(ISO7816.SW_WRONG_P1P2);
  inDataLength =
    nonNegativeByte(buffer[ISO7816.OFFSET_LC]);
  if (inDataLength != 1)
    ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
  receiveIncomingData(apdu);
  short amount = nonNegativeByte(inData[0]);
  if ((amount < 1) || (amount > 100))
    ISOException.throwIt(ISO7816.SW_WRONG_DATA);
  if ((short)(balance + amount) <=  MAX_BALANCE)
    balance = (short)(balance + amount);
  else
    ISOException.throwIt((short)EXCEEDED_BALANCE);
}
...
```

SmartSlang counterpart

```
...
command credit(Amount amount) {
  if (balance + amount <= MAX_BALANCE) {
    balance = balance + amount;
  } else {
    respond EXCEEDED_BALANCE;
  }
} apdu {0x80,0x30,0,0,ubytes[AMOUNT_BYTES](amount),0};
...
```

SmartSlang

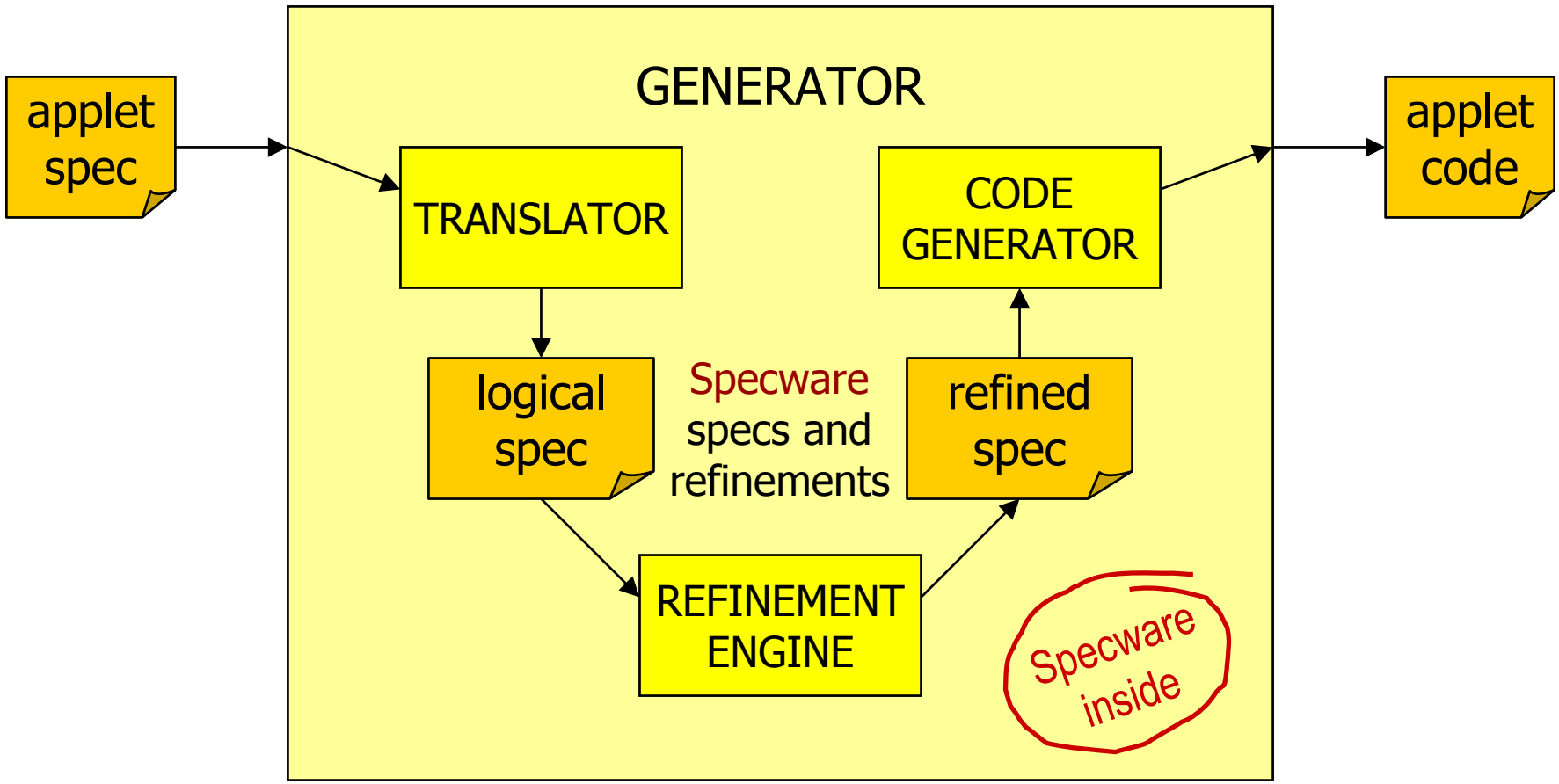
current features not shown in the example

- enumeration types with arguments
- built-in crypto types & functions
- ...

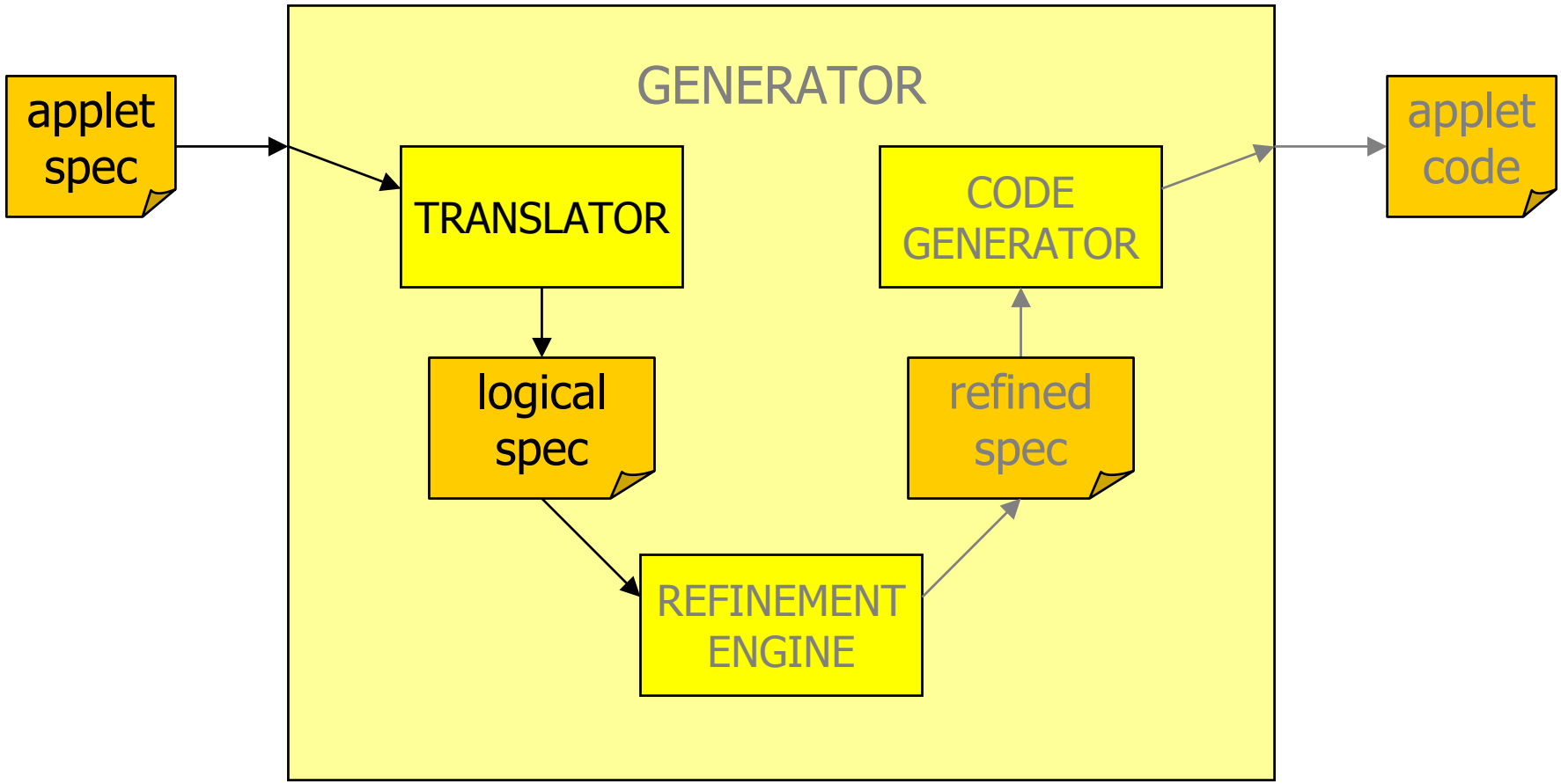
higher-level constructs under design

- PINs
- challenge-response
- multi-command/response exchanges
- FIPS 140-2 [more later ]
- ...

inside the generator



translator



SmartSlang → Specware's specification language (higher-order logic)

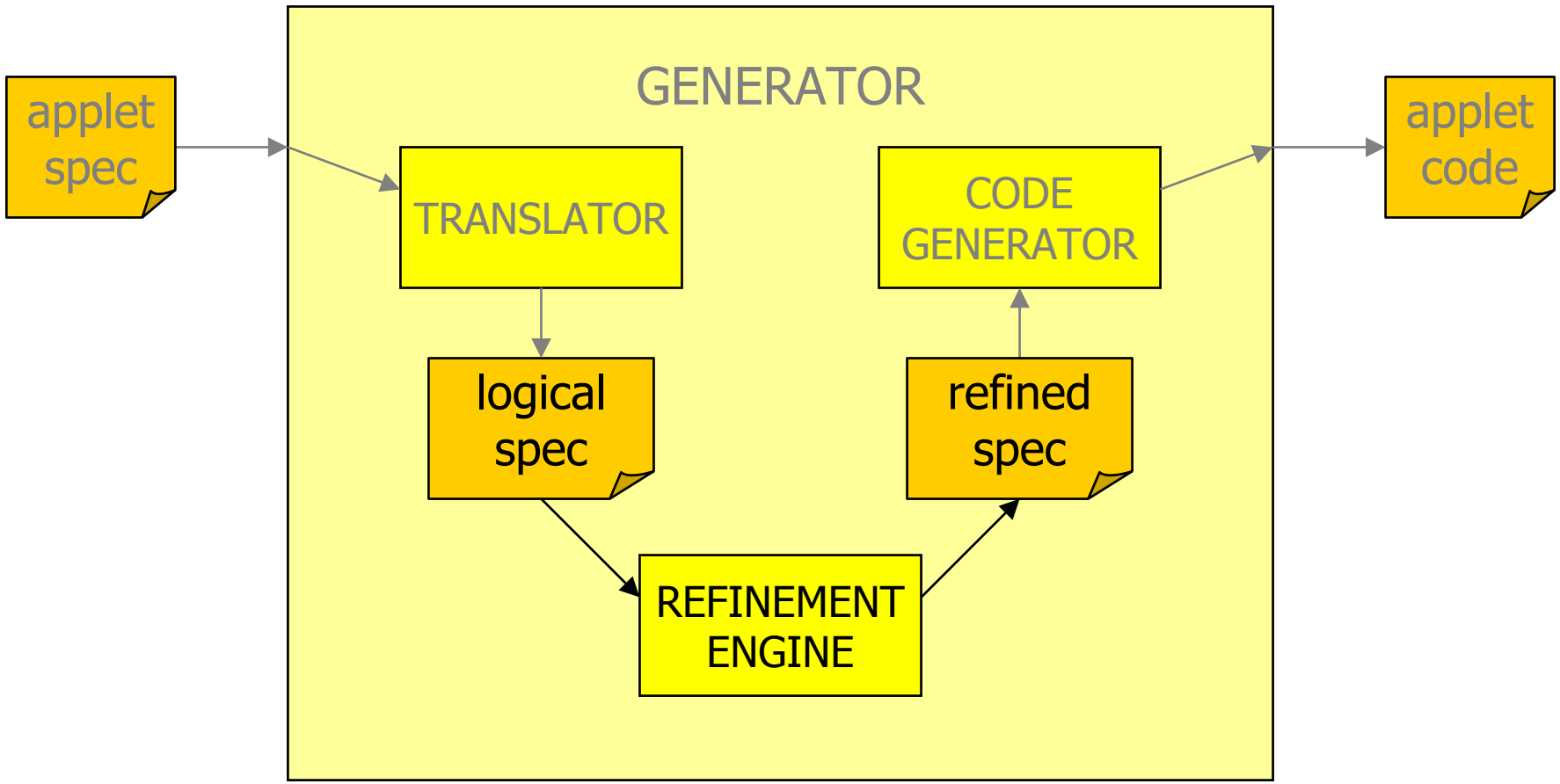
translator

representation
of state machine
higher-order logic

```
sort Command = ...
sort Response = ...
sort State = ...
op applet : Command * State ->
            Response * State
axiom cmd1 is
  fa(s : State) applet(c1,s) = ...
... % more axioms
```

SmartSlang → Specware's specification language (higher-order logic)

refinement engine



applet refinement

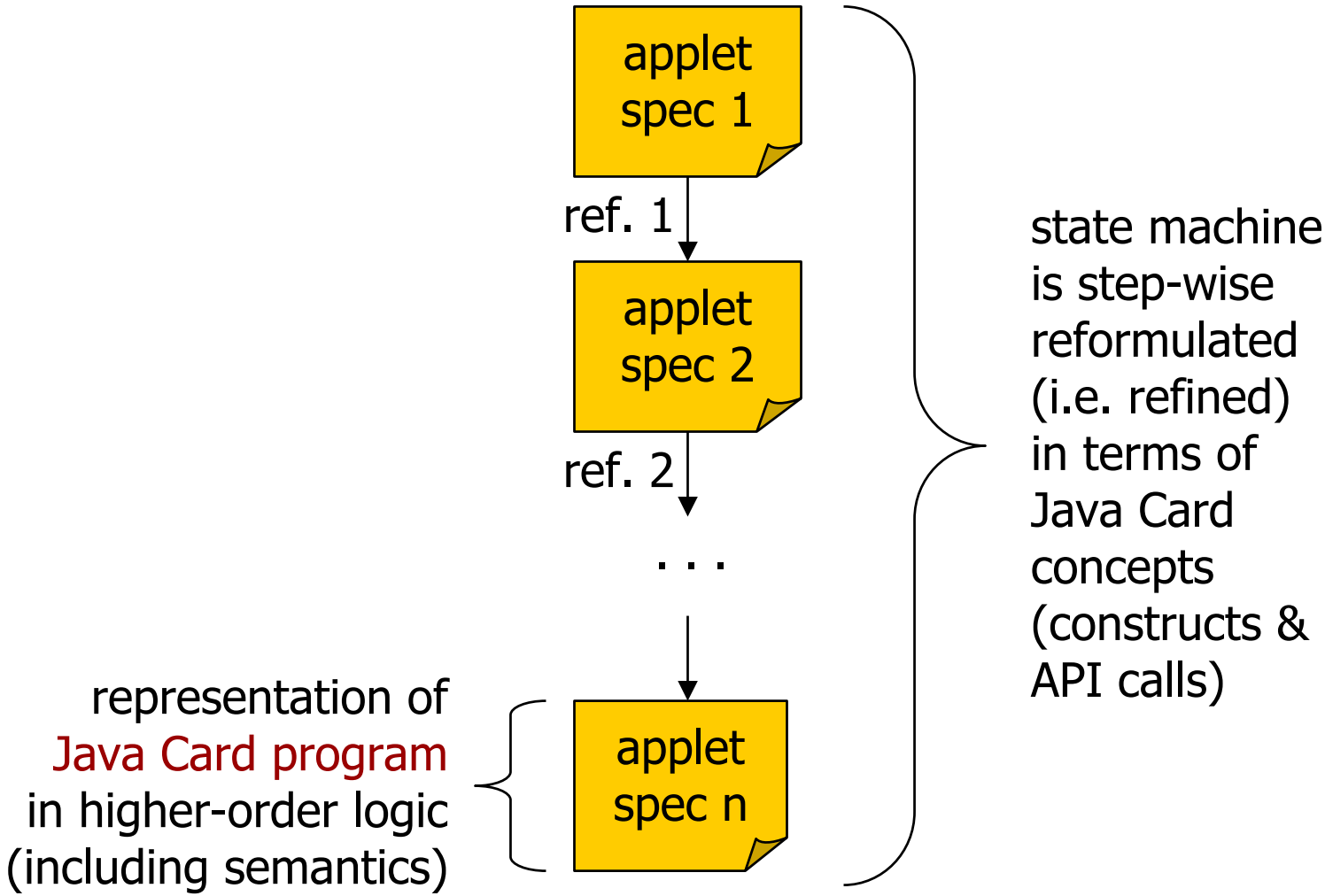
```
...
op applet : Command * State ->
            Response * State
axiom a is ... % more abstract formulation
```

correctness:

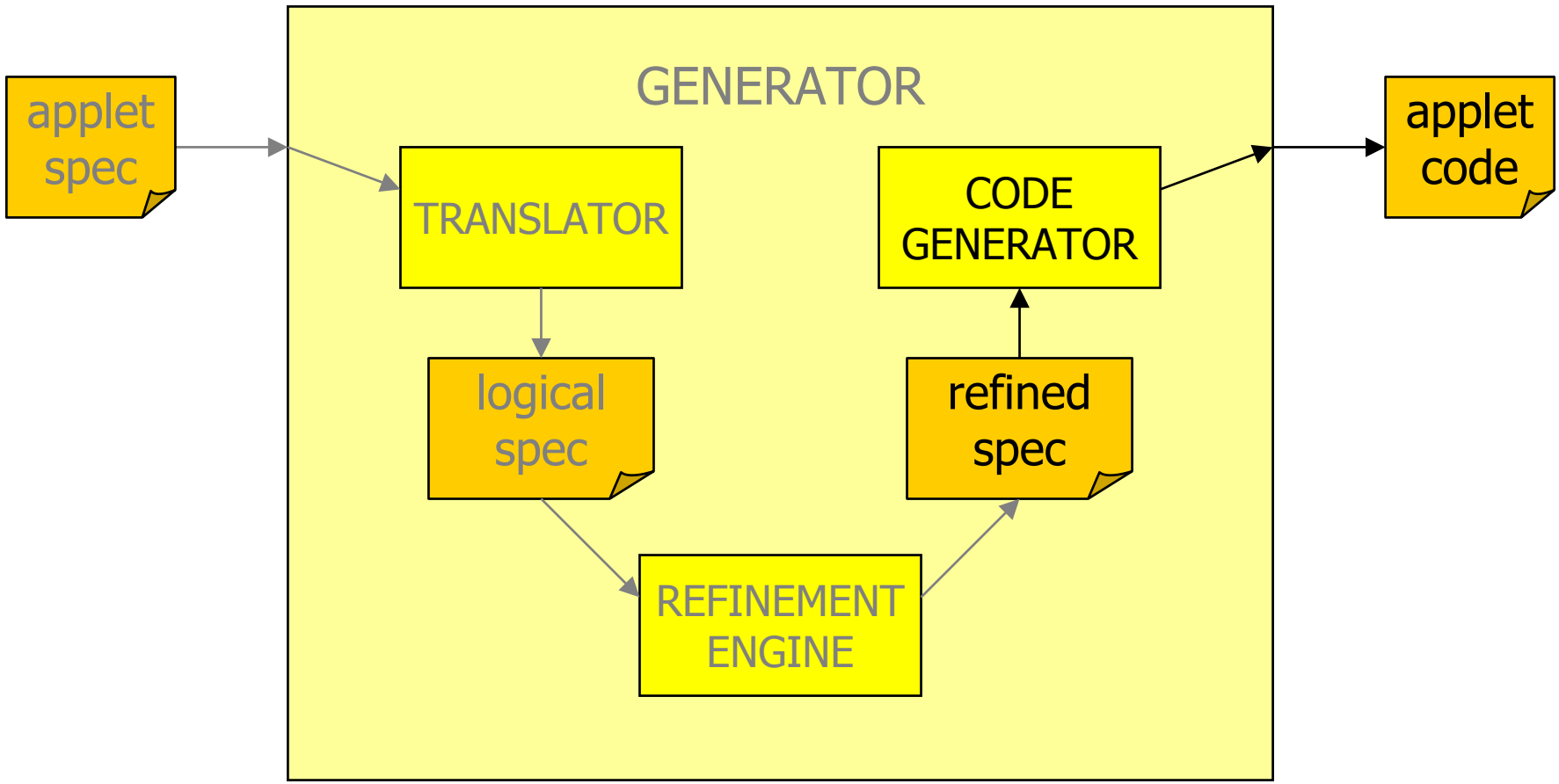
$a' \vdash a$

```
...
op applet : Command * State ->
            Response * State
axiom a' is ... % more concrete formulation
```

applet refinement



code generator



Java Card program representation → Java Card program

code generator

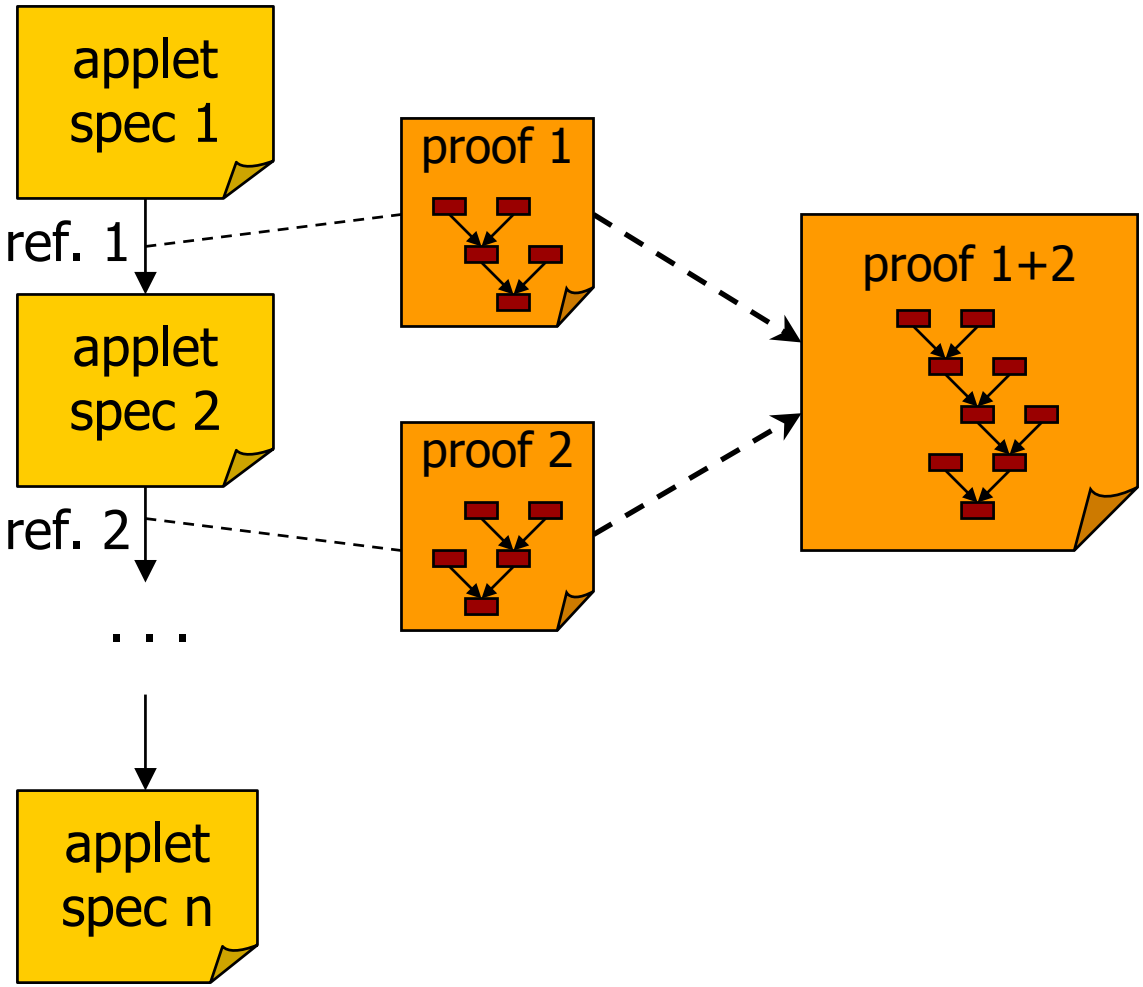
```
sort UserClass = | cls_C
sort UserLocalVariable =
  | locvar_apdu
  | locvar_buf
sort Method = | mth_m
...
axiom
  method_body(mth_m) =
  stat_assignment
    expr_locvar_buf
    (expr_invoke_APDU_getBuffer
       expr_locvar_apdu)
```

```
class C {
 void m(APDU apdu)
 {
  byte[] buf;
  buf =
   apdu.getBuffer();
 }
}
```

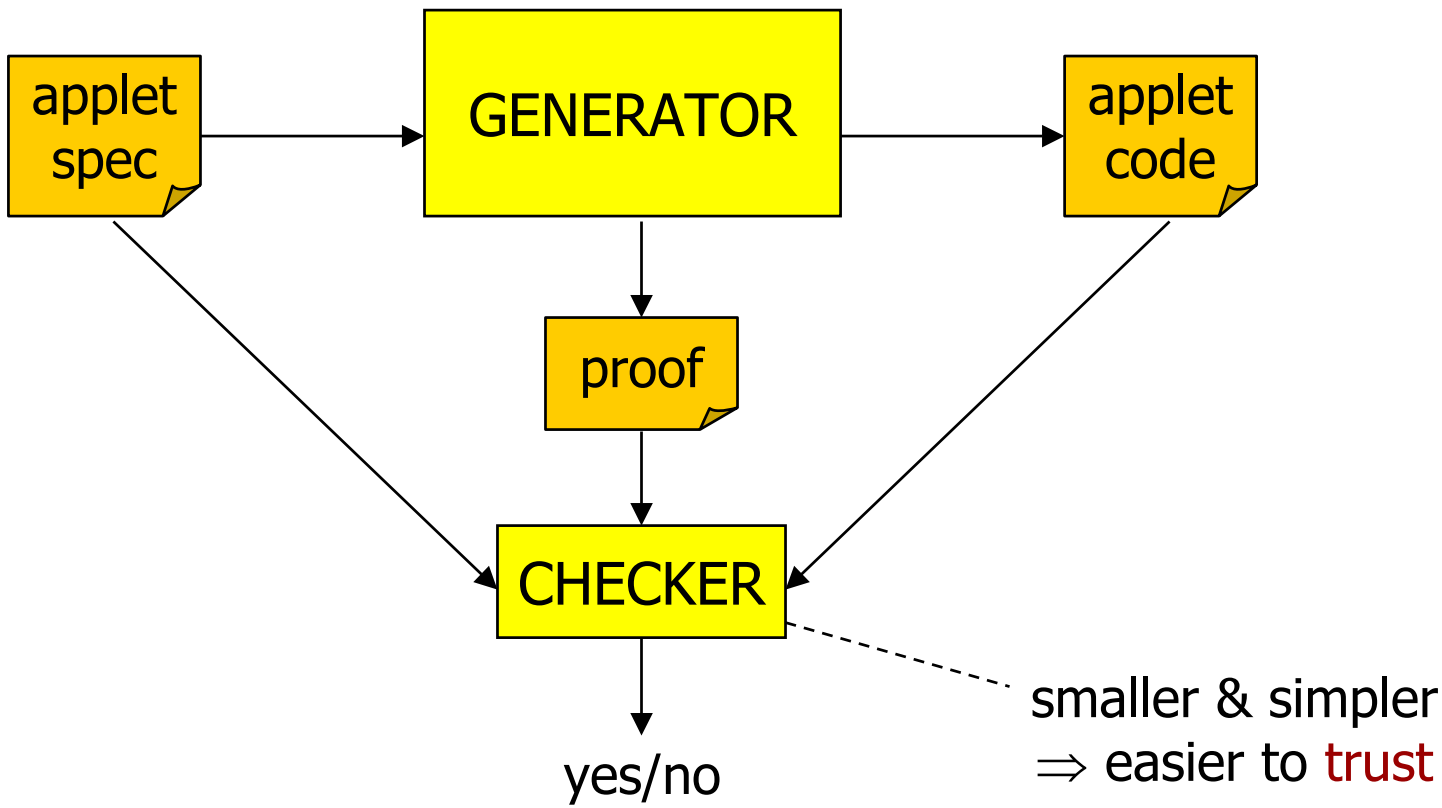
Java Card program representation

Java Card program

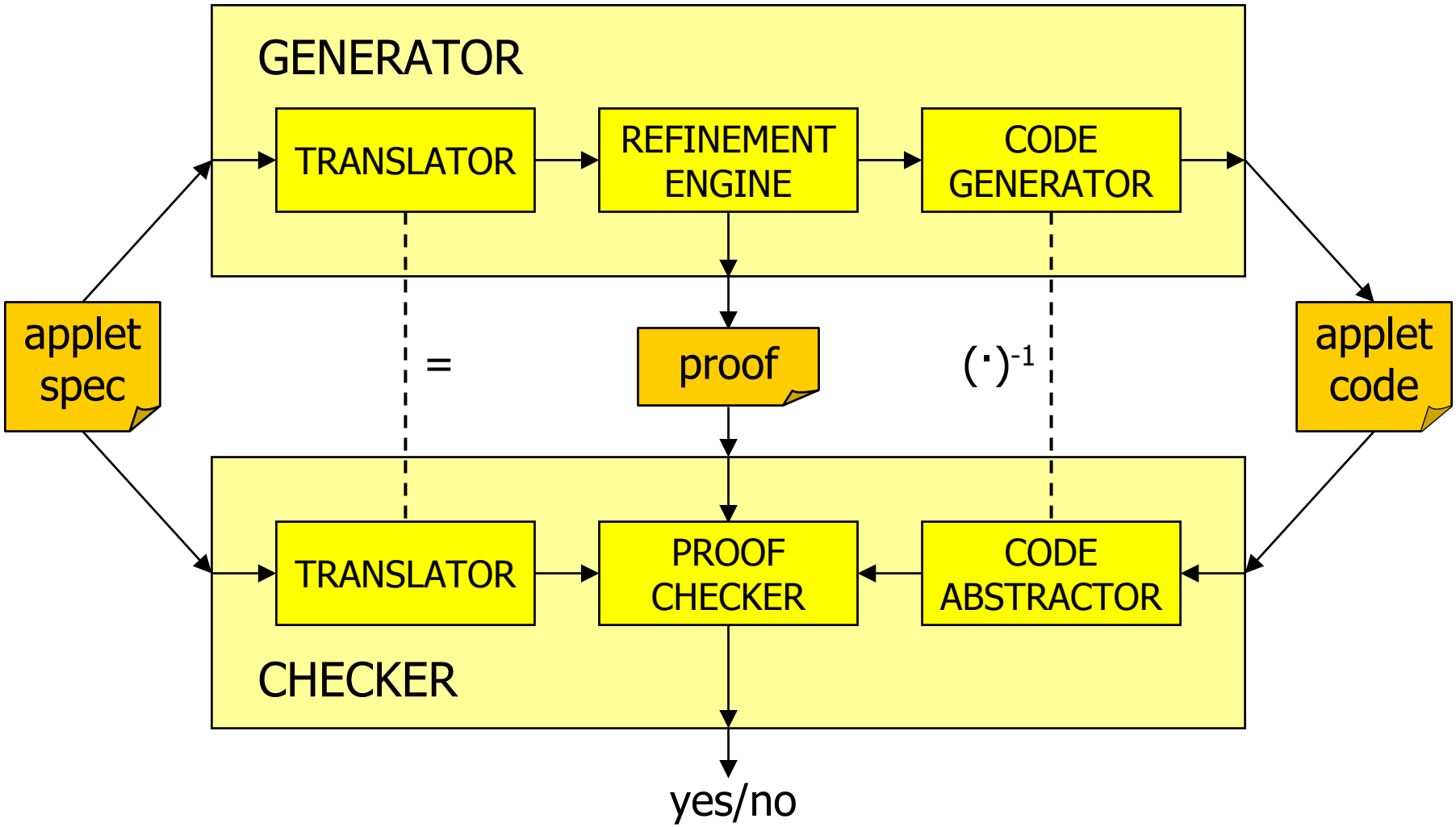
refinement proofs



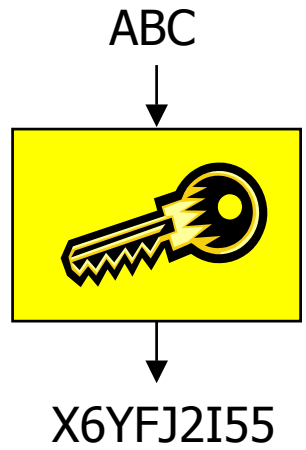
independent certification



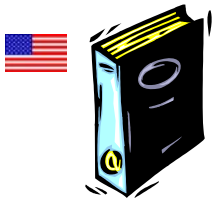
independent certification



FIPS 140-2



federal standard for cryptographic modules



11 areas

4 levels

requirements

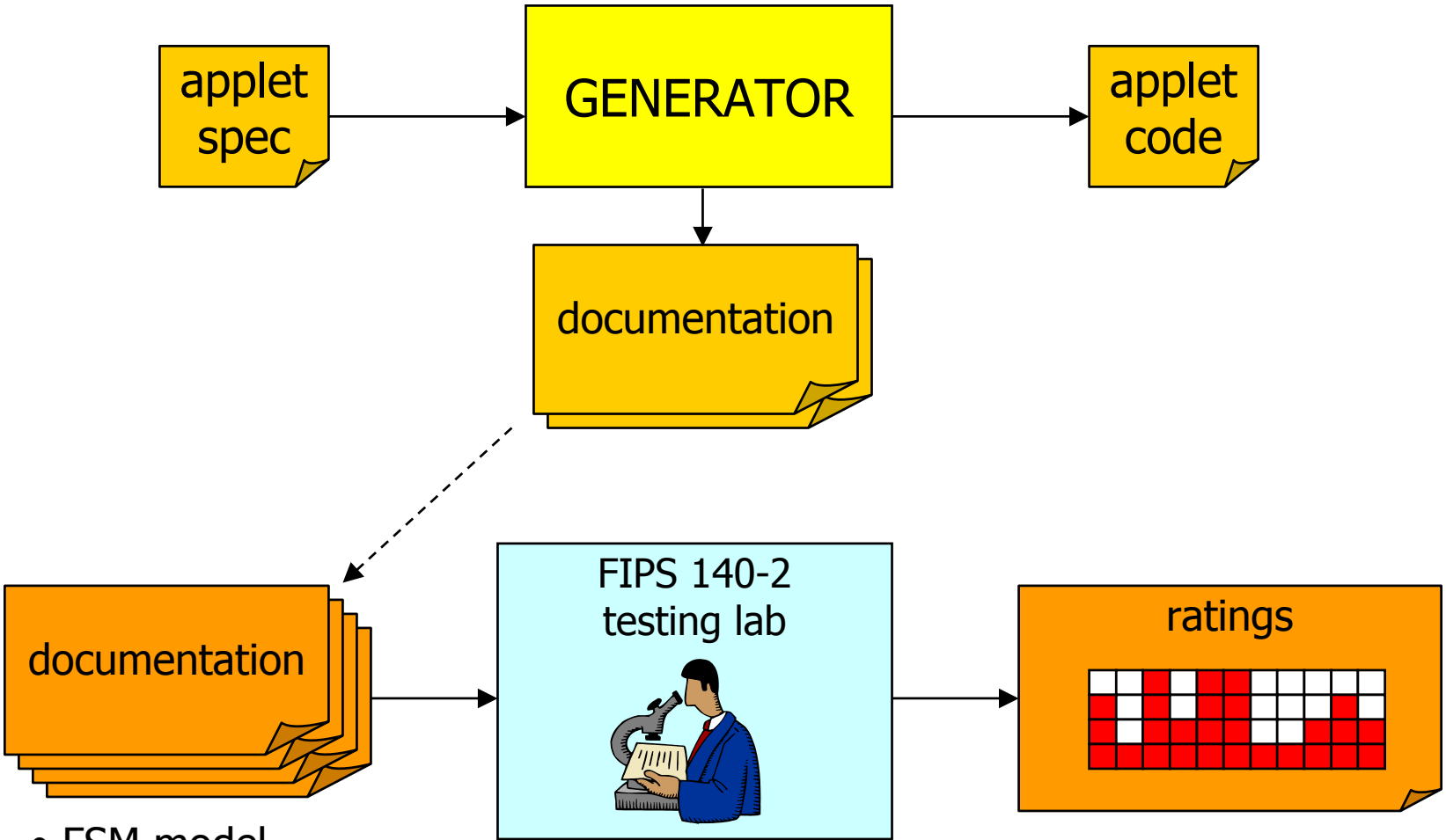
documentation

- FSM model
- security policy
- …

FIPS 140-2
testing lab

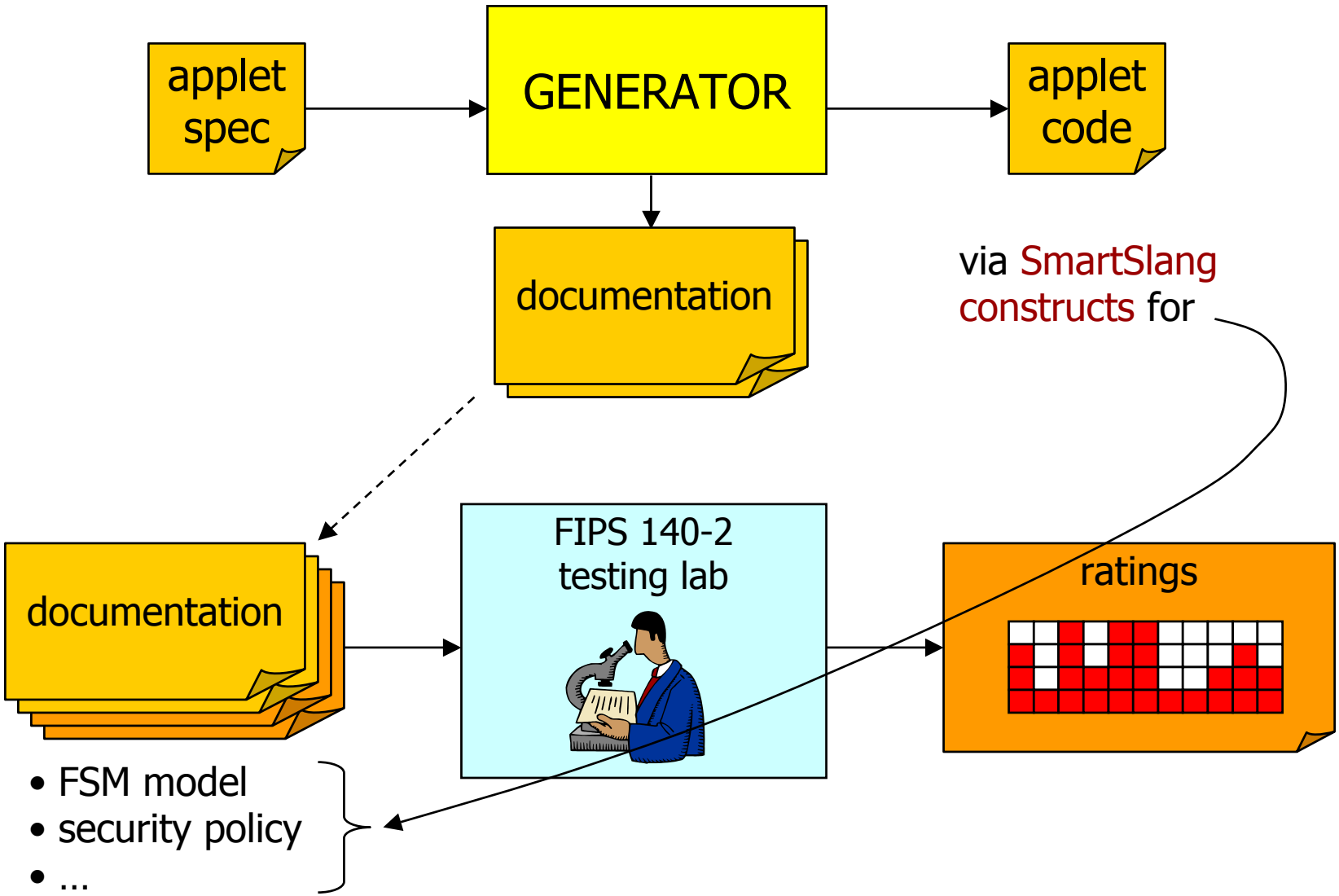
ratings

FIPS 140-2



- FSM model
- security policy
- …

FIPS 140-2

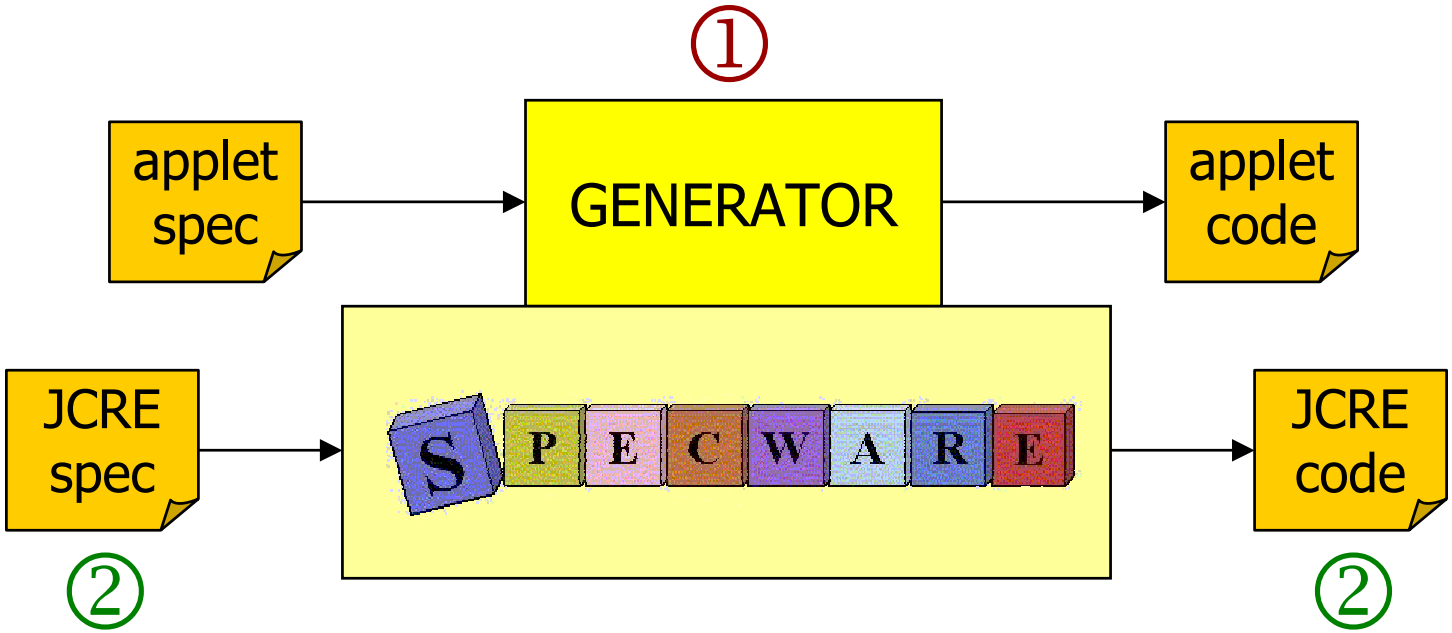

applet spec
GENERATOR
applet code
documentation
via SmartSlang constructs for
documentation
FIPS 140-2 testing lab
ratings
• FSM model
• security policy
• …

ongoing 

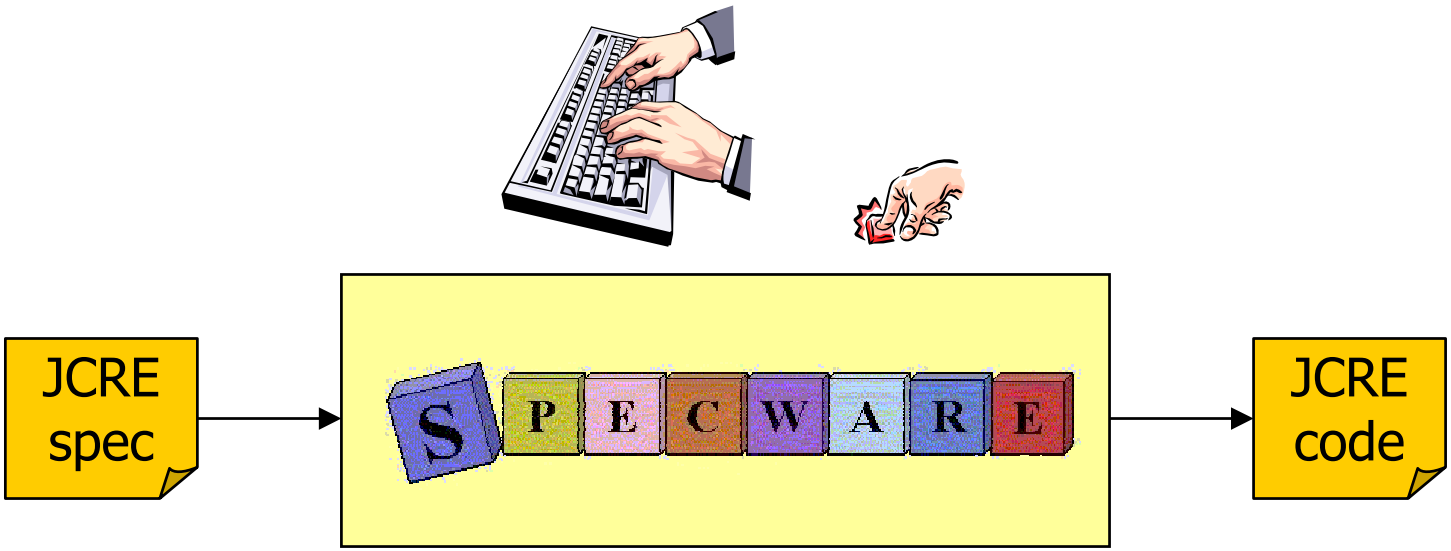
project at Kestrel 

① Java Card applet generator

② synthesis of Java Card Runtime Environment



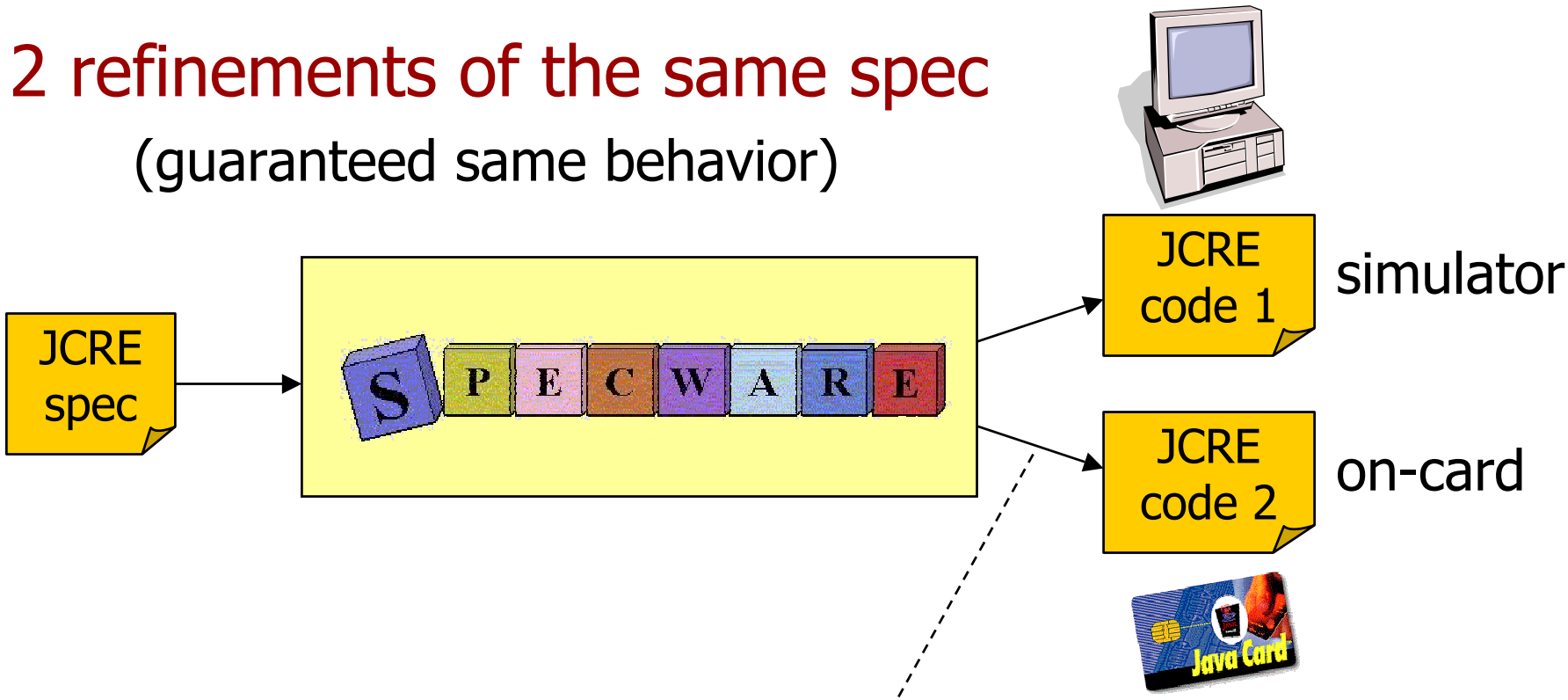
JCRE synthesis



JCRE synthesis

2 refinements of the same spec

(guaranteed same behavior)



examples from JCRE spec

```
sort PrimitiveType = | boolean
                     | byte
                     | short
                     % no int

sort ArrayType = | primarr   PrimitiveType
                 | classarr  Class
                 | interfarr Interface
     % mono-dimensional arrays

sort ReferenceType = | class  Class
                     | interf Interface
                     | array  ArrayType

sort Type = | prim PrimitiveType
            | ref  ReferenceType
```

examples from JCRE spec

```
sort TCNumber = ...  % two's complement numbers

sort ByteValue = {tcn : TCNumber | len tcn = 8}
sort ShortValue = {tcn : TCNumber | len tcn = 16}

sort NonNullReference  % abstract

sort TypedReference = {ref  : NonNullReference,
                       type : ObjectType}

sort ReferenceValue = | nullref
                      | nonnull TypedReference

sort Value = | bool  BooleanValue
             | byte  ByteValue
             | short ShortValue
             | ref   ReferenceValue

op has_type? : Value * Type -> Boolean
```

examples from JCRE spec

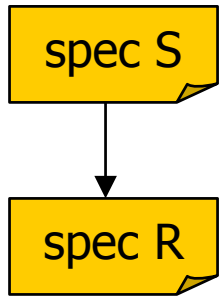
```
sort FieldStore = {fs : FMap(Field,Value) |
                   (fa (f : Field, v : Value)
                       apply fs f = some v =>
                       has_type? (v, field_type f))}

sort ClassInstance = {ci : {class  : Class,
                            fields : FieldStore,
                            iinfo  : InternalClassInfo} |
                       domain ci.fields = owned_fields ci.class &
                       internal_info_class ci.iinfo = ci.class}

sort Array = {a : {type       : ArrayType,
                   components : FSeq Value} |
              fa (v : Value)
                 v in_seq? a.components =>
                 has_type? (v, arrcomptype a.type)}

sort Object = | clinst ClassInstance
              | array  Array
```

examples from JCRE spec

```
sort Heap =
  {hp : FMap(TypedReference,Object) |
   % each reference-object pair is type-consistent:
   (fa (tr : TypedReference, o : Object)
       apply hp tr = some o => tr.type = object_type o) &
   % non-null references in class instances point to objects:
   (fa (tr  : TypedReference,
        ci  : ClassInstance,
        f   : Field,
        tr1 : TypedReference)
       apply hp tr = some(clinst ci) &
       apply (ci.fields) f = some(ref(nonnull tr1)) =>
       tr1 in? domain hp) &
   % non-null references in arrays point to objects:
   (fa (tr  : TypedReference,
        a   : Array,
        tr1 : TypedReference)
       apply hp tr = some(array a) &
       ref(nonnull tr1) in_seq? a.components =>
       tr1 in? domain hp)}
```

proof obligations

- generated from refinements



$a_R \vdash a_S$

- generated from specs

e.g.
```
op change_field : Heap * ... -> Heap
def change_field(hp,...) = ...
```

proof obligation:

$\vdash$ result of `change_field` preserves heap invariants

spec validation
- catch spec errors
- increase confidence in spec

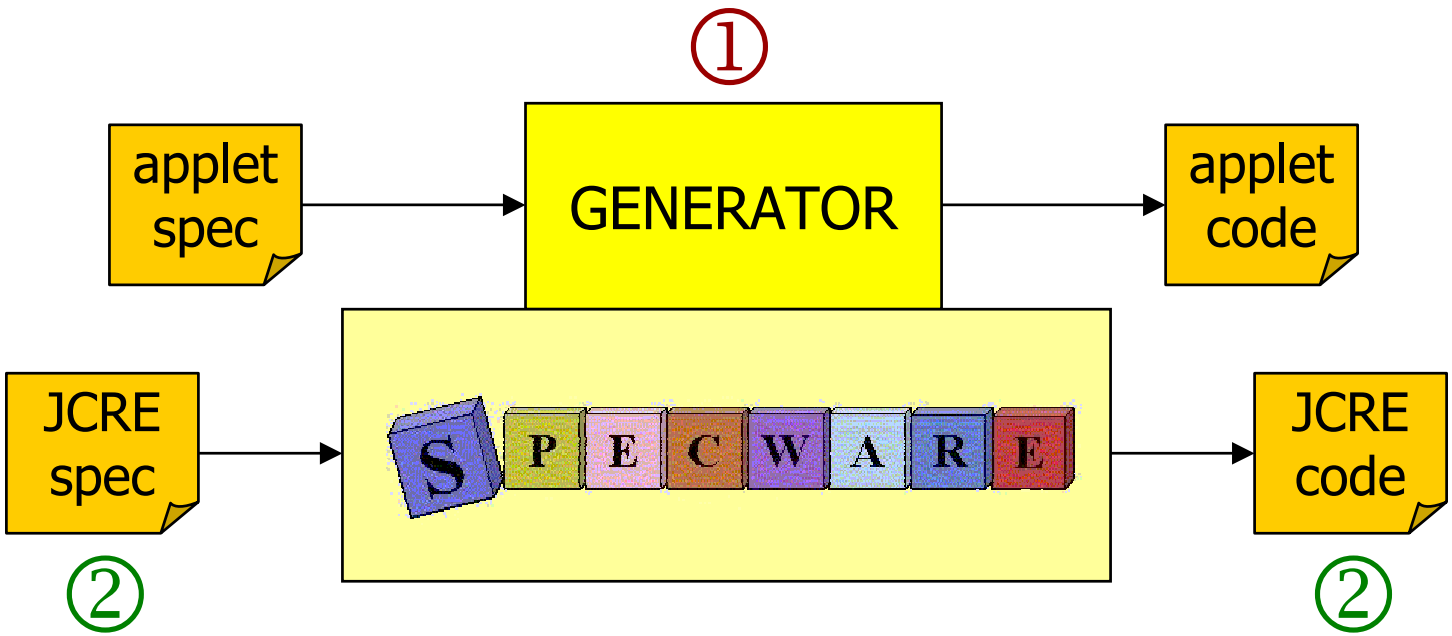
Java Card language (excl. API) spec:

- ~ 800 proof obligations
- ~ 400 proved so far

ongoing  project at Kestrel 

① Java Card applet generator

② synthesis of Java Card Runtime Environment



other contributors

- Matthias Anlauff
- Scott Burson
- Eric Bush
- David Cyrluk
- Li-Mei Gilham
- Jim McDonald
- Lambert Meertens
- Weilyn Pa
- Stephen Westfold

for more information

www.kestrel.edu/java

www.kestrel.edu/jcapplets

www.specware.org