Extensible Record Structures in Event-B

Asieh Salehi Fathabadi, Colin Snook,
Thai Son Hoang, Dana Dghaym and Michael Butler
University of Southampton, UK

ABZ 2021, June 9th 2021 (virtual)

ABZ 2021

UNIVERSITY OF Southampton

Overview

- Motivation
- CamilleX tool
- Record Structure: Syntax and Transformation
- Case study: Tokeneer

Motivation

- standard **Event-B** mathematical language:
  - system state is modelled using data structures
  - no support for the ***direct*** definition of structured types such as ***records***
- **extending** the Event-B language:
  - ***direct*** record definitions
  - new fields in refinement steps
  - more readable models
  - retaining the ease of refinement and proof

CamilleX tool

- Camillex:
  - an extensible **text representation** of Event-B models (Xtext framework)
  - two types of text files: XMachine and XContext

    (an Event-B model contains two parts: contexts for static data and machines for dynamic behaviour)
  - automatically translated to the corresponding Rodin machine or context
  - based on the EMF framework for Event-B
- **extending** CamilleX grammer:
  - support the new **records** extension
  - records are translated to standard Event-B

Record Structure: Syntax

- in an Event-B XMachine or XContext text file:

```
record record_id [extends extended_record_id]
(field_id: [multiplicity] field_type) *
```

- **multiplicity**: min and max number of times the field element:
  - one, opt, many (default: one)
- **extension**: allowing record structures to model hierarchies (refinement)

Record Structure: Semantics

- static record fields are specified in a context
- dynamic record fields are specified in a machine
  - **hierarchical** definition of data structures:
    - a record in a context/machine extends a record specified in the same context/machine
  - both **static** and **dynamic** data:
    - A record in a machine, extends a record in a context seen by the machine
  - data **refinement**:
    - A record in a refining context/machine, extends a record in the abstract context/machine

Record Structure: Transformation

- in a context:

(non-extending record)
sets record_id

(extending record)
constants record_id
axioms
record_id ⊆ extended_record_id

constants field_id
axioms field_id ∈
record_id (/ ⇸ / →) field_type

- in a machine:

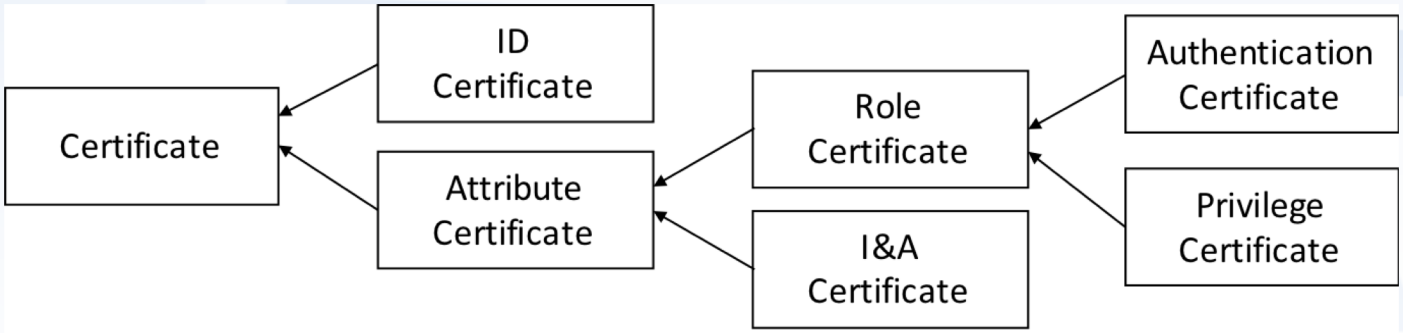
variables record_id
invariants
record_id ⊆ extended_record_id

variables field_id
invariants field_id ∈
record_id (/ ⇸ / →) field_type

Case Study: Tokeneer*

allow only authorised users access to the enclave

- hierarchy of tokeneer certificate types:



* **HiClass** project (113213), which is part of the ATI Programme, a joint Government and industry investment to maintain and grow the UK's competitive position in civil aerospace design and manufacture.

```
context c2_card extends c1_door
sets KEYPART PRIVILEGE
        CLEARANCE TOKENID
        FINGERPRINT
records
record CERTIFICATE
    idIssuer : issuer
    validityPeriod : time
    signature : opt KEYPART
record IDCert extends CERTIFICATE
    subject : USER
    publicKey : KEYPART
record AttCert extends CERTIFICATE
    baseCertId : issuer
    tokenId : TOKENID
record RoleCert extends AttCert
    role : PRIVILEGE
    clearance : CLEARANCE
record PrivCert extends RoleCert
record AuthCert extends RoleCert
record IandACert extends AttCert
    fingerprintTemplate : FINGERPRINT
record TOKEN
    tokenID : TOKENID
    idCert : IDCert
    privCert : PrivCert
    iandACert : IandACert
record CARD
    token : TOKEN
record USER extends USER
    fingerprint : FINGERPRINT
end
```

```
machine m2_card refines m1_door sees
        c2_card
variables validToken
records
record USER extends USER
    holds : opt CARD
record TOKEN extends TOKEN
    authCert : opt AuthCert
invariants
@inv1 : validToken ⊆ TOKEN
@inv2 : holds~ ∈ CARD ⇸ USER
@inv3 : ∀tkn . tkn ∈ validToken ⇒
    baseCertId(privCert(tkn)) = idIssuer(
        idCert(tkn)) ∧
    baseCertId(iandACert(tkn)) =
        idIssuer(idCert(tkn)) ∧
    tokenId(privCert(tkn)) = tokenID(tkn
        ) ∧
    tokenId(iandACert(tkn)) = tokenID(
        tkn)
events
event holdCard any user crd where
  @grd1 : user ∈ USER
  @grd2 : crd ∈ CARD
  @gdr3 : user ∉ dom(holds)
  @grd4 : crd ∉ ran(holds)
  @gdr5 : token(crd) ∈ validToken
  @gdr6 : fingerprint(user) =
        fingerprintTemplate(iandACert(
        token(crd))) then
  @act1 : holds(user) := crd end
end
```

ABZ 2021

UNIVERSITY OF Southampton

Thank you

Questions?