Data-types definitions: Use of Theory and Context instantiations Plugins[1]

Peter Rivière[†], Neeraj Kumar Singh[†], Yamine Ait-Ameur[†]

[†] INPT-ENSEEIHT / IRIT Université de Toulouse, France

June 8, 2021

9th Rodin User and Developer Workshop

INP TOULOUSE ENSEEIHT ABZ'2021 IRIT

[1] This work is supported by ANR-EBRP project.

Outline

Context and Objectives

Context

Context instantiation work is in porgress in EBRP ANR project[2].

Interest

- Access to Rodin Tools e.g. ProB model checker/animator [LB03] to be used for the development of theory-based Event-B models.
- Improve proof automation.
- Keep the syntactic constructs of the Theory-PlugIn

Objectives

Providing a set of transformation rules of theories [BM10] into Event-B contexts [Abr10].

[2] https://www.irit.fr/EBRP

Overview

Theory	Context
THEORY Th	CONTEXT Ctx
IMPORT Th1, ...	SETS s
TYPE PARAMETERS E, F, ...	CONSTANTS c
DATATYPES	AXIOMS A
Type1$(E, ...)$	THEOREMS T_{ctx}
constructors	END
cstr1$(p_1 : T_1, ...)$	
OPERATORS	
Op1 <nature> $(p_1 : T_1, ...)$	
well–definedness $WD(p_1, ...)$	
direct definition D_1	
AXIOMATIC DEFINITIONS	
TYPES A_1, ...	
OPERATORS	
AOp2 <nature> $(p_1 : T_1, ...) : T_r$	
well–definedness $WD(p_1, ...)$	
AXIOMS A_1, ...	
THEOREMS T_1, ...	
PROOF RULES	
REWRITE RULES	
INFERENCE RULES	
END	
(a)	(b)

Table: Global structure of Event-B Theories and Context

Data Types

- Some properties are implicit in data type definitions
- Need of an explicit transformation rule

```
THEORY
    Data_Type_Schema

TYPE PARAMETERS
    T1,T2

DATATYPES
  Struct :
        //base case 1
    cons1
        //base case 2
    cons2(el1:T1, el2:T2)
```

CONTEXT
Data_Type_Schema

SETS
T1, T2, Struct

CONSTANTS
cons1, cons2, cons2Type, el1, el2

AXIOMS

axm1: $Partition(Struct, \{cons1\}, cons2Type)$

axm2: $cons2 \in T1 \times T2 \rightarrowtail\!\!\!\!\rightarrow cons2Type$

axm3: $el1 \in cons2Type \rightarrow T1$

axm4: $el2 \in cons2Type \rightarrow T2$

Operators : Direct definitions

- Direct Definition of an operator ⟶ lambda expression
- Predicate operators ⟶ BOOL type expression

```
THEORY
    Direct_Expr_Schema

TYPE PARAMETERS
    T1,T2

OPERATORS

    opE <expression >
        (arg1 : T1, arg2 : T2)
      well−definedness  WD1,WD2
      direct definition
        Exp(arg1, arg2)

    opP <predicate>
        (arg1 : T1, arg2 : T2)
      well−definedness  WD1,WD2
      direct definition
        PExpr(args1, arg2)
```

```
CONTEXT
    Direct_Expr_Schema

SETS
    T1, T2

CONSTANTS
    opE, opP

AXIOMS
axmE:  opE = (λargs1 ↦ args2·
               args1 ∈ T1 ∧ args2 ∈ T2∧
               WD1 ∧ WD2 |
               Exp(args1, args2)

axmP:  opP = (λargs1 ↦ args2·
               args1 ∈ T1 ∧ args2 ∈ T2∧
               WD1 ∧ WD2 |
               bool(PExpr(args1, args2)))
```

Operators : Axiomatic definitions

- Axiom in a theory ⟶ Axiom in the context
- Typing axioms are added to support well-defined condition

```
THEORY
Axm_Schema

TYPE PARAMETERS
T1 , T2

AXIOMS OPERATORS
    op <expression>
        (arg1 : T1, arg2 : T2) 8 Res_type
      well−definedness WD1,WD2

AXIOMS
    axm1 : Exp1(op,...)
    ...
    axmn : Expn(op,...)
```

```
CONTEXT
    Axm_Schema

SETS
    T1 , T2

CONSTANTS
    op

AXIOMS
    axmDefOp : op ∈
      {args1 ↦ args2 · arg1 ∈ T1 ∧ arg2 ∈ T2 ∧ WD1 ∧ WD2}
                → Res_type
    axm1 : Exp1(op,...)
    ...
    axmn : Expn(op,...)
```

Example: Sequence

```
THEORY
Seq_Theo

TYPE PARAMETERS
A

OPERATORS

  seq <expression> (a: ℙ(A))
  direct definition
    {f,n · n ∈ ℕ ∧ f ∈ 1..n→a|f}

  seqSize <expression> (s: ℤ↔A)
  well−definedness
    s ∈ seq(A)
  direct definition
        card(s)
END
```

```
CONTEXT
  Seq_CTX

SETS
  A

CONSTANTS
  seq
  seqSize

AXIOMS

  axm2:
    seq = (λa · a ∈ ℙ(A) | {f, n · n ∈ ℕ ∧ f ∈ 1..n → a | f})
  axm4:
    seqSize = (λs · s ∈ ℤ ↔ A ∧ s ∈ seq(A) | card(s))
END
```

Inductive Schema

- Rely on recursive functions schemas of J.R Abrial and D. Cansell
- Instantiation of the specific FrSB function schema

CONTEXT SchemaRecGen
SETS S_type , B_type
CONSTANTS wellfounded , fix , FrSB , S, B
AXIOMS

axm0: $S \subseteq S\_type$

axm6: $B \subseteq B\_type$

@axm1: $wellfounded = \{r \cdot r \in S \leftrightarrow S \wedge (\forall p \cdot p \subseteq S \wedge p \subseteq r[p] \Rightarrow p = \emptyset\}$

@axm2: $fix \in (\mathbb{P}(S \times B) \rightarrow \mathbb{P}(S \times B)) \rightarrow \mathbb{P}(S \times B)$

@axm3: $\forall h \cdot h \in \mathbb{P}(S \times B \rightarrow \mathbb{P}(S \times B) \wedge (\forall a, b \cdot a \subseteq b \wedge b \subseteq S \times B \Rightarrow h(a) \subseteq h(b)) \Rightarrow fix(h) = h(fix(h))$

axm4: $FrSB = \{r \mapsto g \mapsto fr \mid r \in wellfounded \wedge g \in S \times (S \nrightarrow B) \nrightarrow B \wedge (\forall x, f \cdot x \in S \wedge f \in S \nrightarrow B \wedge r \sim [\{x\}] \subseteq dom(f) \Rightarrow x \mapsto f \in dom(g)) \wedge fr = fix(\lambda p \cdot p \in S \leftrightarrow B \mid \{x, h \cdot x \in S \wedge r \sim [\{x\}] \lhd h \subseteq p \wedge r \sim [\{x\}] \lhd h \in r \sim [\{x\}] \rightarrow B \mid x \mapsto g(x \mapsto r \sim [\{x\}] \lhd h)\})\}$

lem1: $FrSB \in \{r \mapsto g \mid r \in wellfounded \wedge g \in S(S \nrightarrow B) \nrightarrow B \wedge (\forall x, f \cdot x \in S \wedge f \in S \nrightarrow B \wedge r \sim [\{x\}] \subseteq dom(f) \Rightarrow x \mapsto f \in dom(g))\} \rightarrow (S \leftrightarrow B)$

THEOREMS

thm1: $\forall r, g, fr \cdot r \mapsto g \mapsto fr \in FrSB \Rightarrow fr \in S \rightarrow B$

thm2: $\forall r, g, fr \cdot r \mapsto g \mapsto fr \in FrSB \Rightarrow (\forall x \cdot x \in S \Rightarrow fr(x) = g(x \mapsto r \sim [\{x\}] \lhd fr))$

Induction type definition (Step 1)

```
THEORY Ind_Data_Type_Schema
TYPE PARAMETERS T
DATATYPES
 IndType :
  cons1
          //base case 1
  cons2 (el:T)
          //base case 2
  consinduc1(el :IndType)
          // inductive case 1
  consinduc2(el1 : T, el2
       IndType)
          // inductive case 2
```

```
CONTEXT Ind_Data_Type_Schema
SETS IndTypeTYPE , T
CONSTANTS IndType , IndTypeSET , cons1 , cons2 ,
      consinduc1 , consinduc2
AXIOMS
```

axm1: $IndTypeSET =$

$$\{indtype\_El \mapsto cons1\_El \mapsto cons2\_El \mapsto consinduc1\_El \mapsto consinduc2\_El \mid$$
$$indtype\_El \subseteq IndTypeTYPE \land$$
$$cons1\_El \in indtype\_El \land$$
$$cons2\_El \in T \rightarrowtail (indtype\_El \setminus (ran(consinduc1\_El) \cup ran(consinduc2\_El) \cup \{cons1\_El\})) \land$$
$$consinduc1\_El \in indtype\_El \rightarrowtail (indtype\_El \setminus (ran(cons2\_El) \cup ran(consinduc2\_El) \cup \{cons1\_El\})) \land$$
$$consinduc2\_El \in T \rightarrowtail (indtype\_El \setminus (ran(consinduc1\_El) \cup ran(cons2\_El) \cup \{cons1\_El\})) \land$$
$$(\forall tr \cdot cons1\_El \in tr \land cons2\_El[T] \subseteq tr \land consinduc1\_El[tr] \subseteq tr \land consinduc2\_2[T \times tr] \subseteq tr \Rightarrow indtype\_El \subseteq tr)\}$$

axm2: $IndType \mapsto cons1 \mapsto cons2 \mapsto consinduc1 \mapsto consinduc2 \in IndTypeSET$

END

Operator : Recursive definition (Step 2)

```
THEORY
TYPE PARAMETERS T1,T2
OPERATORS
 op <expression> ( arg1 : T1,arg2 : T2)
 well−definedness WD1,WD2...
 recursive definition
  case cons1 = Exp1(···) //base case 1
  case cons2 = Exp2(···) //base case 2
  case consinduc1 = ExpInd1(op,···)  // inductive case 1
  case consinduc2 = ExpInd2(op,···) // inductive case 2
```

```
CONTEXT
INSTANCIATES SchemaRecGen Ind_Data_Type_Schema
SETS T,T1,T2
CONSTANTS op
AXIOMS
```

$axmn$: $op = \mathrm{FrSB}(\{e \mapsto ind\_el \mid e \in IndType \wedge ind\_el \in IndType \wedge$
$(\exists el \cdot el \in T \wedge ind\_el = consinduc1(el \mapsto e) \vee (\exists el \cdot el \in T \wedge ind\_el = consinduc2(el \mapsto e))\} \mapsto$
$\{e \mapsto f \mapsto res \mid e \in IndType \wedge \boldsymbol{f \in \{ind\_el \mid ind\_el \in IndType \wedge WD1 \wedge WD2\} \rightarrow Res\_Type}$

$(\forall el, ind\_el \cdot el \in T \wedge ind\_el \in IndType \wedge$
$(e = consinduc1(el \mapsto ind\_el) \vee\ e = consinduc2(el \mapsto ind\_el)) \Rightarrow ind\_el \in \boldsymbol{dom(f)}) \wedge$

$(e = cons1(\dots) \Rightarrow res = Exp1(\dots) \vee e = cons2(\dots) \Rightarrow res = Exp2(\dots)) \wedge$

$((\exists el, ind\_el \cdot el \in T \wedge ind\_el \in IndType \wedge e = consinduc1(el \mapsto ind\_el) \Rightarrow res = ExpInd1(f, \dots)) \vee$
$(\exists el, ind\_el \cdot el \in T \wedge ind\_el \in IndType \wedge e = consinduc2(el \mapsto ind\_el) \Rightarrow res = ExpInd2(f, \dots)))\})$

```
END
```

Example

```
THEORY
TYPE PARAMETERS T,S
DATA TYPES
  List(T)
 constructors
     nil()
     cons(head:T, tail:List(T))
OPERATORS
  listSize <expression> (l: List(T))
  recursive definition
     case l
     listSize(nil ) ≙ 0
     listSize(cons(h, q)) ≙ 1 + listSize(q)
END
```

```
CONTEXT List
SETS T List_type
CONSTANTS nil , fix , List_struct , cons ,
      wellfounded , FrSB , listsize , List
AXIOMS
  @axm1,@axm2,@axm3 // fixpoint |
    // S:=List_type ;; S_type:=List_type |thm1,
          thm4, thm5
  axm2: List_struct = {l ↦ n ↦ c | l ⊆ List_type∧
          n ∈ l ∧ c ∈ T × l ↣ l \ {n}∧
          l = fix(λtr · tr ∈ ℙ(List_type) | {n} ∪ c[T × tr])}

  @axm4,@axm5,@axm6,@axm7 // SchemaRecGen |
    // B:=ℕ ;; S:=List_type ;; S_type:=List_type ;;
          B_type:=ℤ| @axm1, lem1, thm1, thm2
  axm3: List ↦ nil ↦ cons ∈ List_struct
  axm4: listsize = FrSB({q ↦ l |
        q ∈ List_type ∧ l ∈ List_type∧
        (∃h · h ∈ T ∧ l = cons(h ↦ q))} ↦
        {l ↦ f ↦ n | l ∈ List_type ∧ f ∈ List_type ⇸ ℕ∧

        (∀h, q · h ∈ T ∧ q ∈ List_type ∧ l = cons(h ↦ q)
            ⇒ q ∈ dom(f))∧

        (l = nil ⇒ n = 0)∧

        (∃h, q · h ∈ T ∧ q ∈ List_type ∧ l = cons(h ↦ q)
            ⇒ n = 1 + f(q))})
END
```

Conclusion

- A systematic transformation of Theories as Event-B contexts
- Study automatisation and develop a tool
- Proof rules can be transformed as theorems, to be proved, in Event-B Contexts
- Add destructor operators for inductive types

Bibliography

[Abr10] Jean-Raymond Abrial. *Modeling in Event-B: system and software engineering*. Cambridge University Press, 2010.

[BM10] Michael Butler and Issam Maamria. Mathematical extension in Event-B through the rodin theory component. 2010.

[LB03] Michael Leuschel and Michael Butler. Prob: A model checker for b. In *International symposium of formal methods europe*, pages 855–874. Springer, 2003.