UNIVERSITY OF Southampton
School of Electronics and Computer Science

Class Diagrams for Abstract Data Types

Thai Son Hoang, Colin Snook, Dana Dghaym, and Michael Butler

(accepted for ICTAC 2017)

ECS, University of Southampton, U.K.

Event-B Mini-Symposium

ECS, University of Southampton, 27th June 2017

Outline

UNIVERSITY OF Southampton
School of Electronics and Computer Science

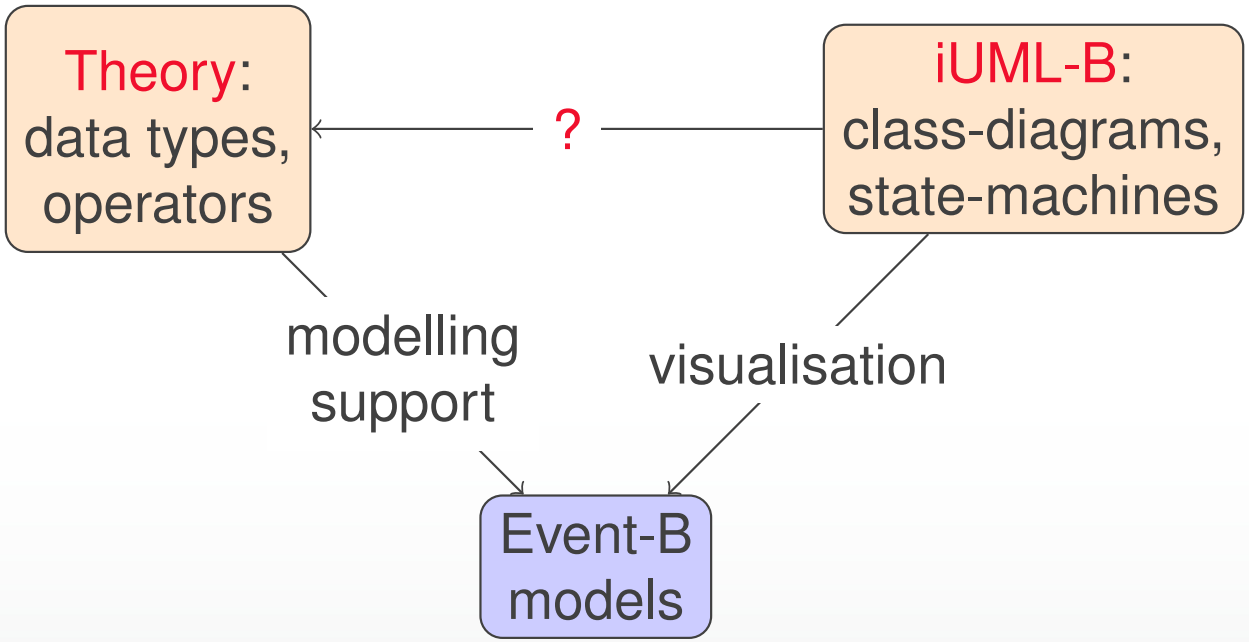
Motivation

Illustration. The Stack ADT

Example. The RailGround Case Study

Motivation. Visualisation for ADTs

UNIVERSITY OF Southampton
School of Electronics and Computer Science



The Aim

Aiding design of ADTs using class diagrams.

Illustration. Stack ADT

UNIVERSITY OF Southampton
School of Electronics and Computer Science

Stack ADT

- Polymorphic datatype.
- `emptyStack`: denotes an empty stack.
- `top`: takes a non-empty stack `st` and returns `st`'s top element.
- `pop`: takes a non-empty stack `st` and returns a stack where `st`'s top element is removed.
- `push`: takes a stack `st` and an element `e` and returns a stacks where `e` is added to the top of `st`.

Illustration. Stack ADT

UNIVERSITY OF Southampton
School of Electronics and Computer Science

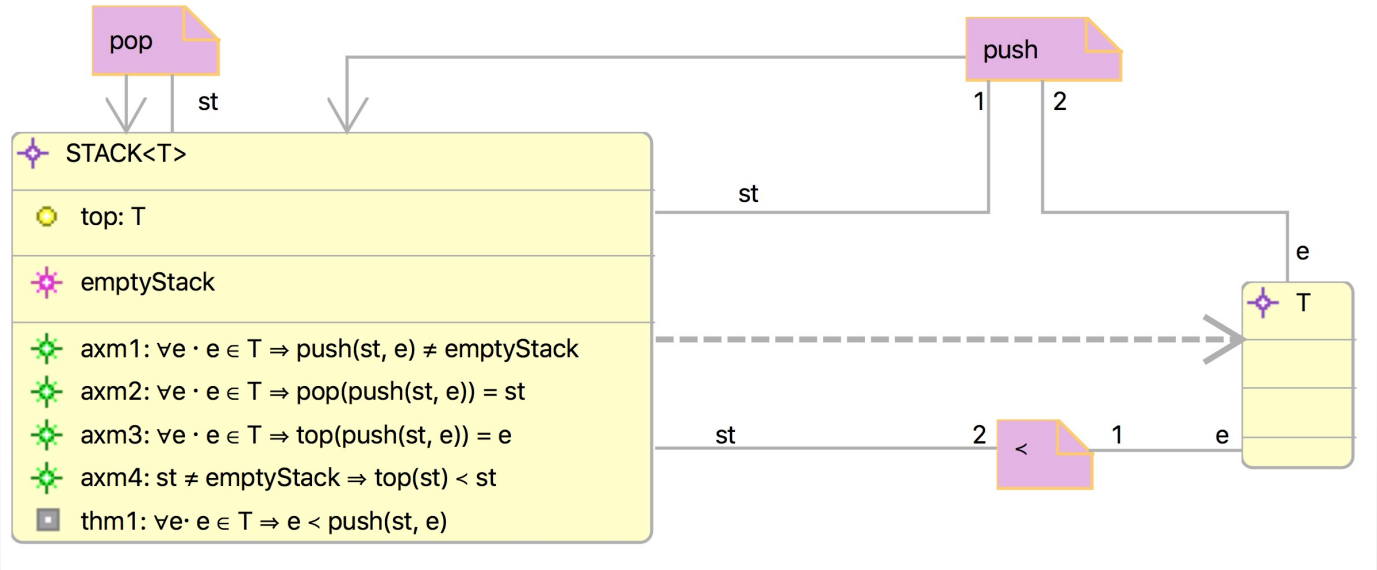
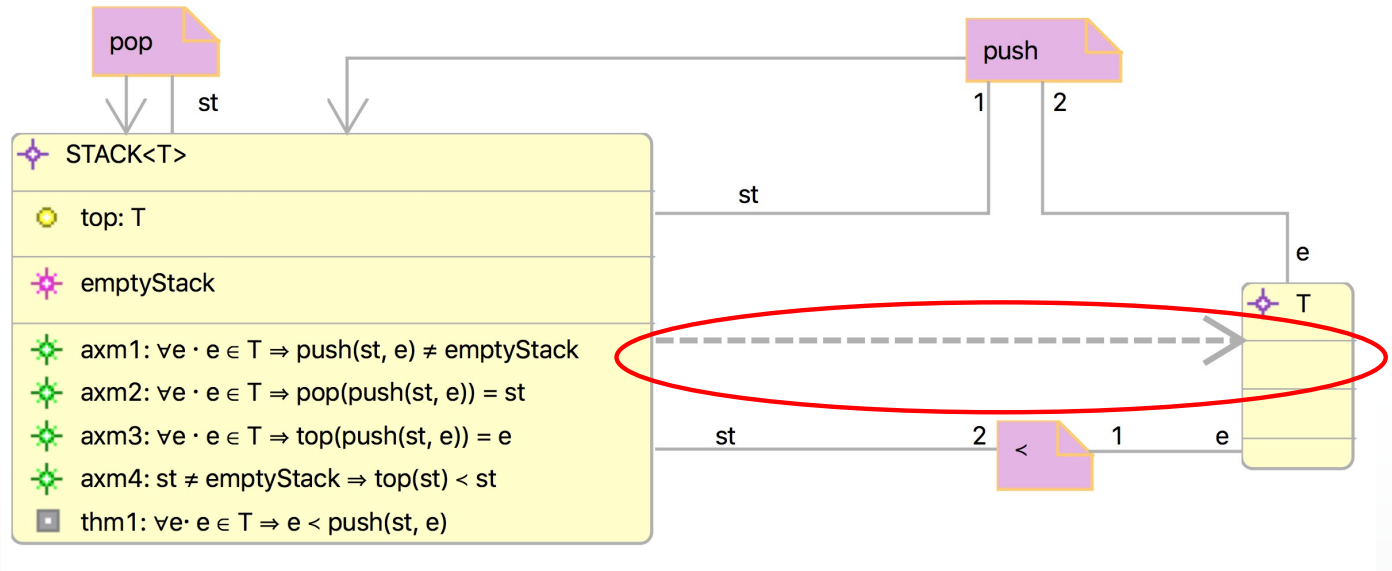


Illustration. Stack ADT

UNIVERSITY OF Southampton
School of Electronics and Computer Science

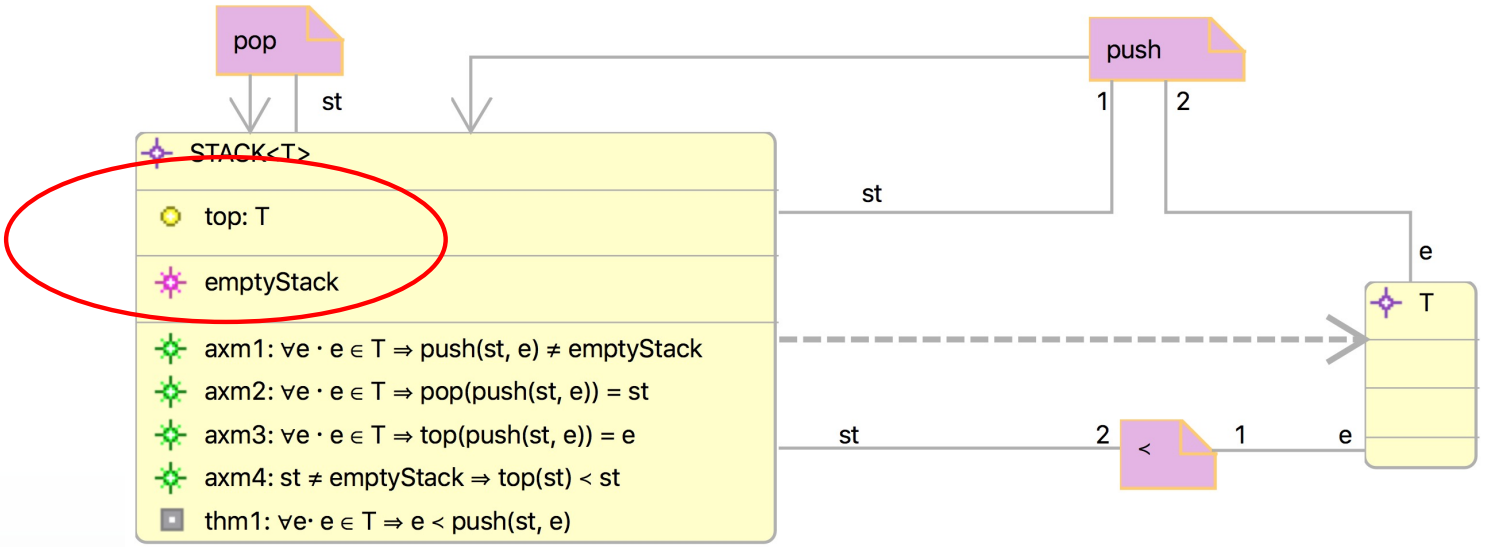


- `STACK` is polymorphic with `T` is the type parameter.

```
1 theory STACK<T>
```

Illustration. Stack ADT

UNIVERSITY OF Southampton
School of Electronics and Computer Science

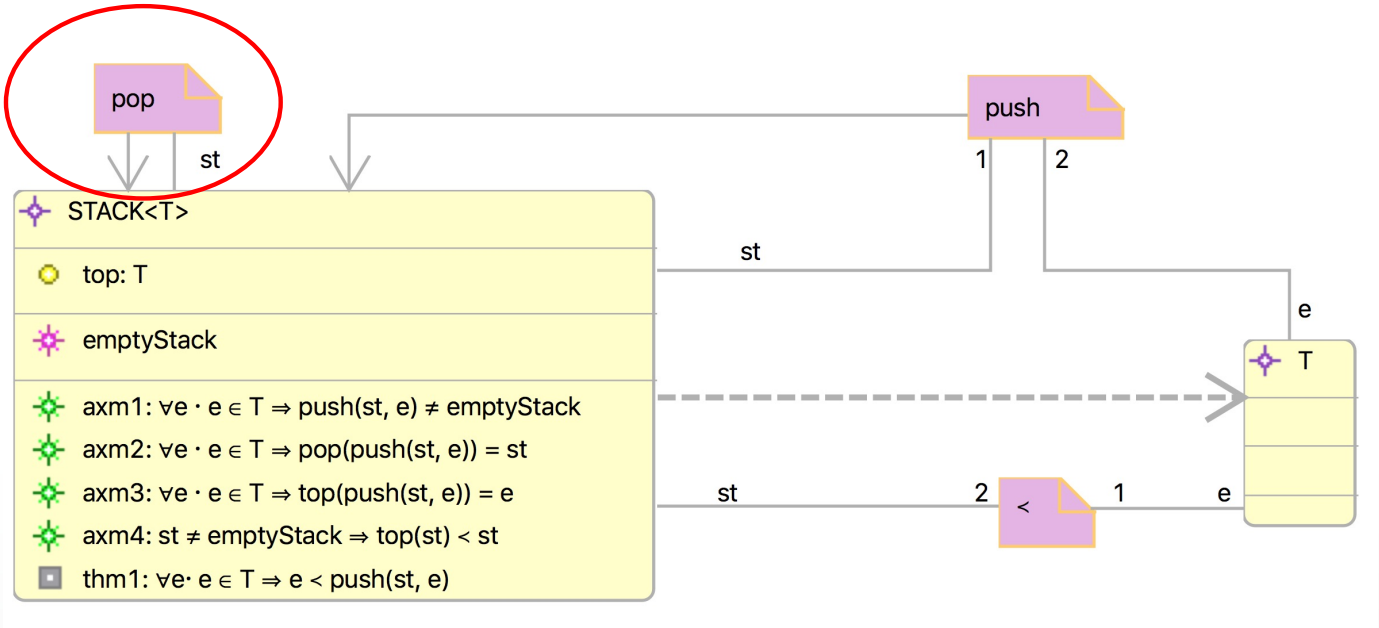


- `emptyStack` is constant `STACK`
- `top` is a "query" operator returning an element of type `T`

```
1  emptyStack: "STACK<T>"
2  top(st : "STACK<T>"): "T"
```

Illustration. Stack ADT

UNIVERSITY OF Southampton
School of Electronics and Computer Science

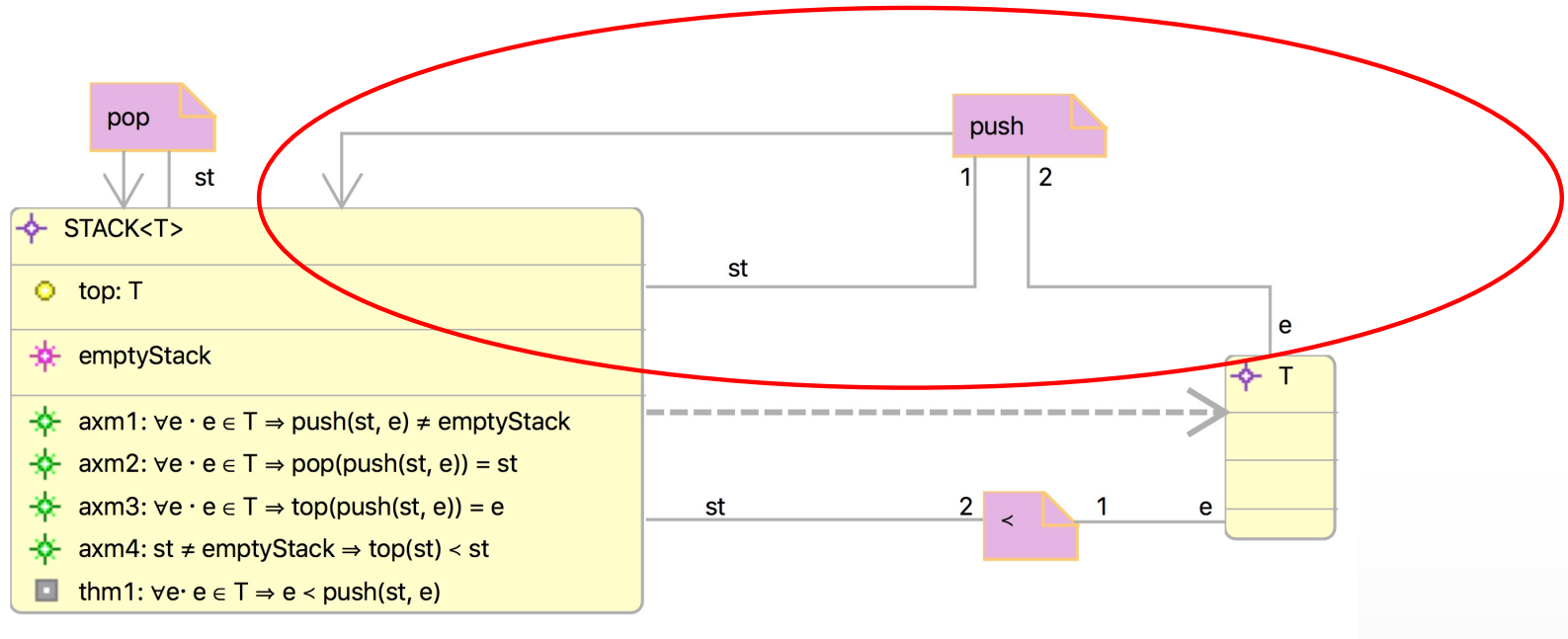


- `pop` is an operator with one input (`st`) and one output (of type `STACK<T>`)

```
1   pop(st: "STACK<T>"): "STACK<T>"
```

Illustration. Stack ADT

UNIVERSITY OF Southampton
School of Electronics and Computer Science

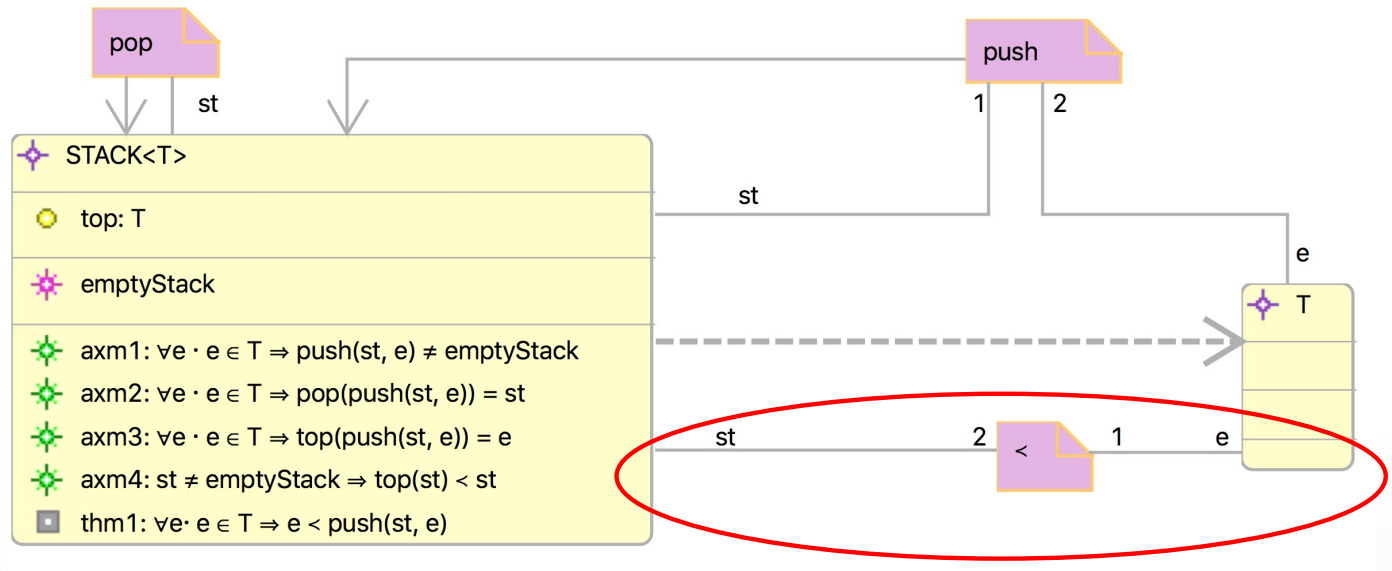


- `push` is an operator with two inputs (`st` and `e`) and one output (of type `STACK<T>`)

```
1   push(st: "STACK<T>", e: "T"): "STACK<T>"
```

Illustration. Stack ADT

UNIVERSITY OF Southampton
School of Electronics and Computer Science

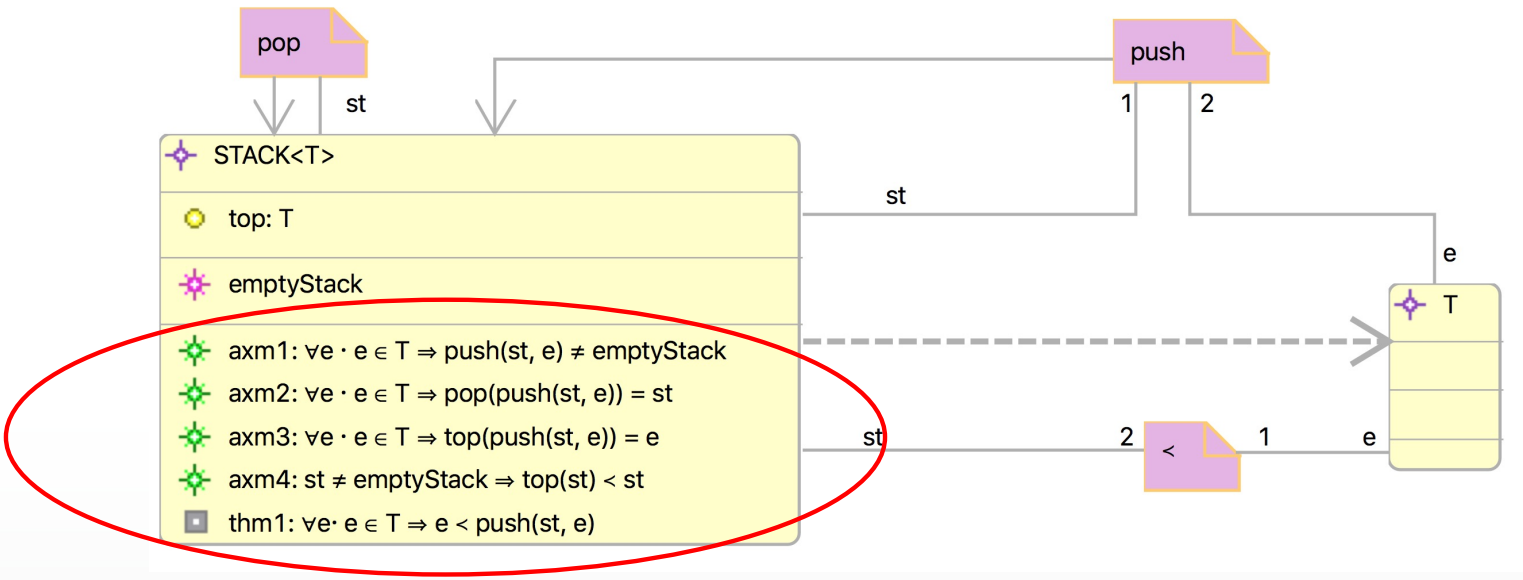


- $\prec$ is a predicate with two inputs (`e` and `st`) stating that `e` is in the stack `st`.

```
1   ≺ (e: "T", st: "STACK<T>") infix
```

Illustration. Stack ADT

UNIVERSITY OF Southampton
School of Electronics and Computer Science



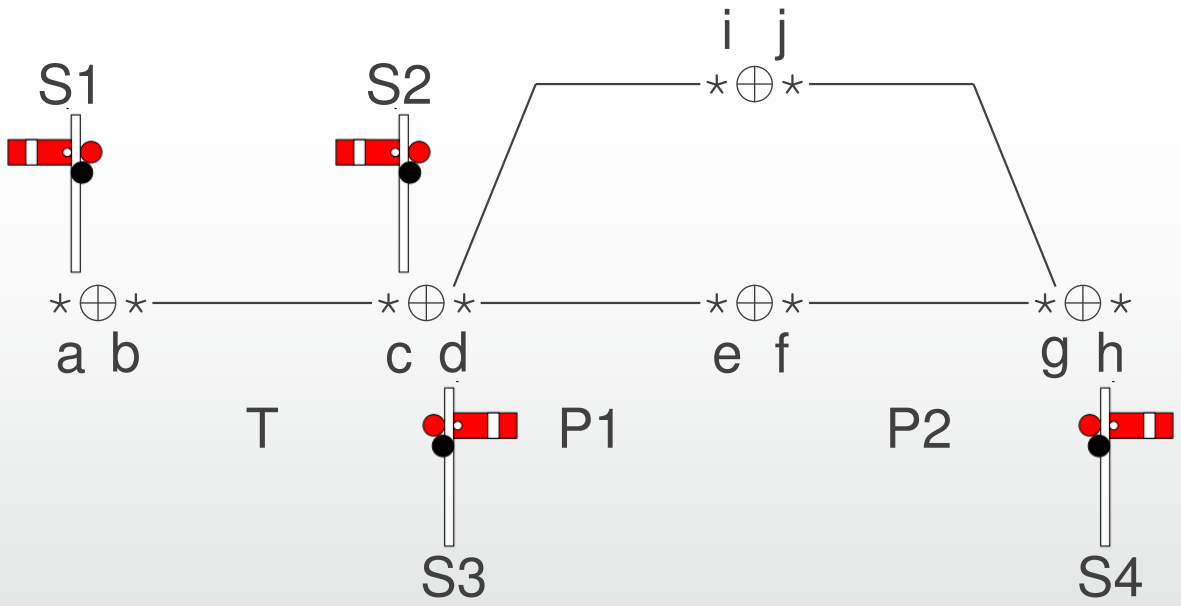
- Axioms are lifted to all instances of the `STACK` datatype, e.g.,

```
1 @axm1: "∀ st·  st ∈ STACK ⇒
2         (∀ e · e ∈ T ⇒ push(st, e) ≠ emptyStack)"
```

RailGround

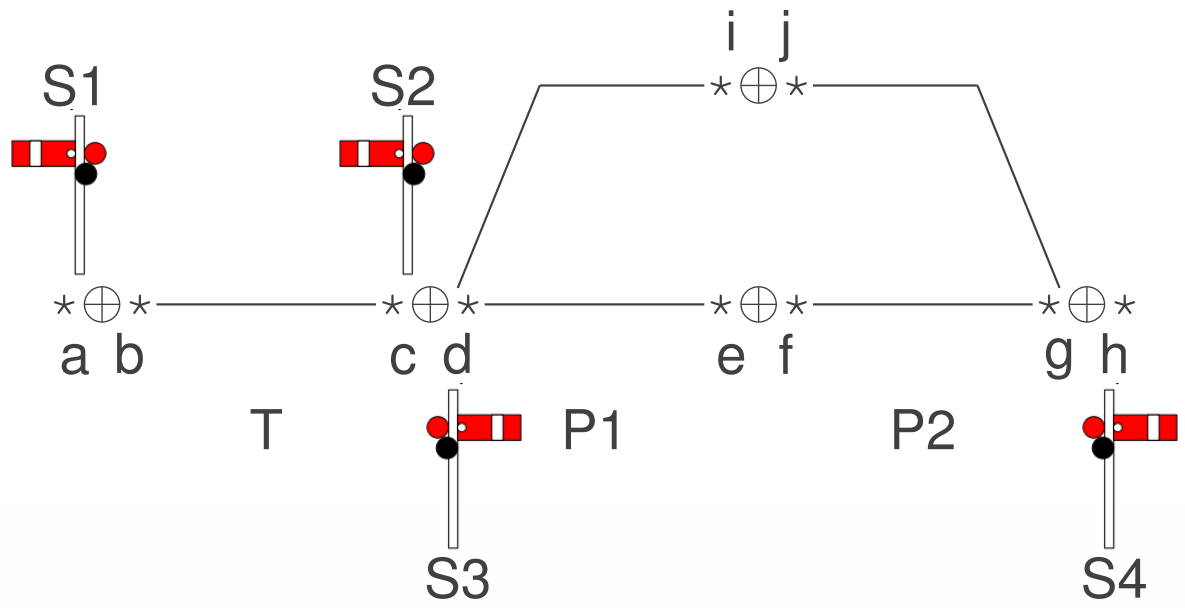
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- A formal model of a (simplified) railway interlocking system.
- Provided by Thales Austria GmbH.
- Research on formal validation and verification of railway systems.



RailGround. Concepture Entities

UNIVERSITY OF Southampton
School of Electronics and Computer Science



- Rail Elements: `T`, `P1`, `P2`
- Rail Connectors: `a, b, c, ...`
- Segments: `bc, de, di, ...`
- Paths/Route: `[bc,de,fg], [bc, di,ig], ...`
- Vacancy Detection: correspond to a set of segments
- Signals: `S1, S2, S3, S4`

Refinement Strategy

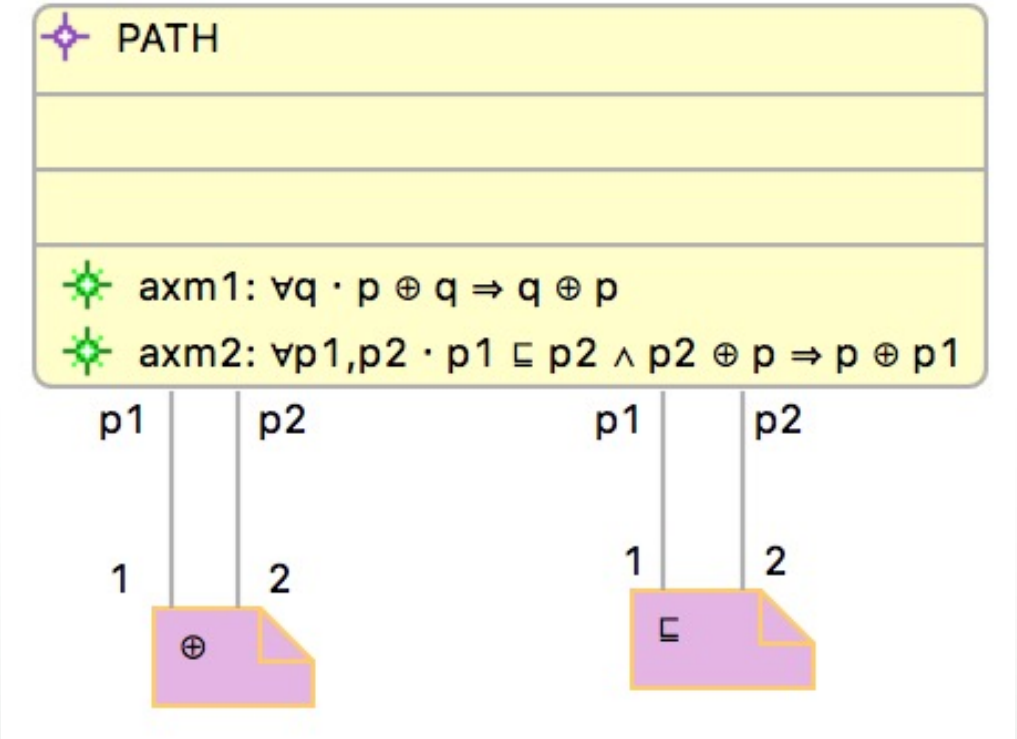
UNIVERSITY OF Southampton
School of Electronics and Computer Science

- **M0**: To abstractly specify active routes in the system, focusing on the collision-free property
- **M1**: To introduce the life-cycle of routes by specifying requested routes.
- **M2**: To formalise the rail elements and the link between rail elements and paths.
- **M3**: To specify the element positions and their association with the rail elements.
- **M4**: To introduce the track vacancy detection mechanism.
- **M5**: To introduce the signals protecting the trains' movement.

Development of Theories

Paths

UNIVERSITY OF Southampton
School of Electronics and Computer Science

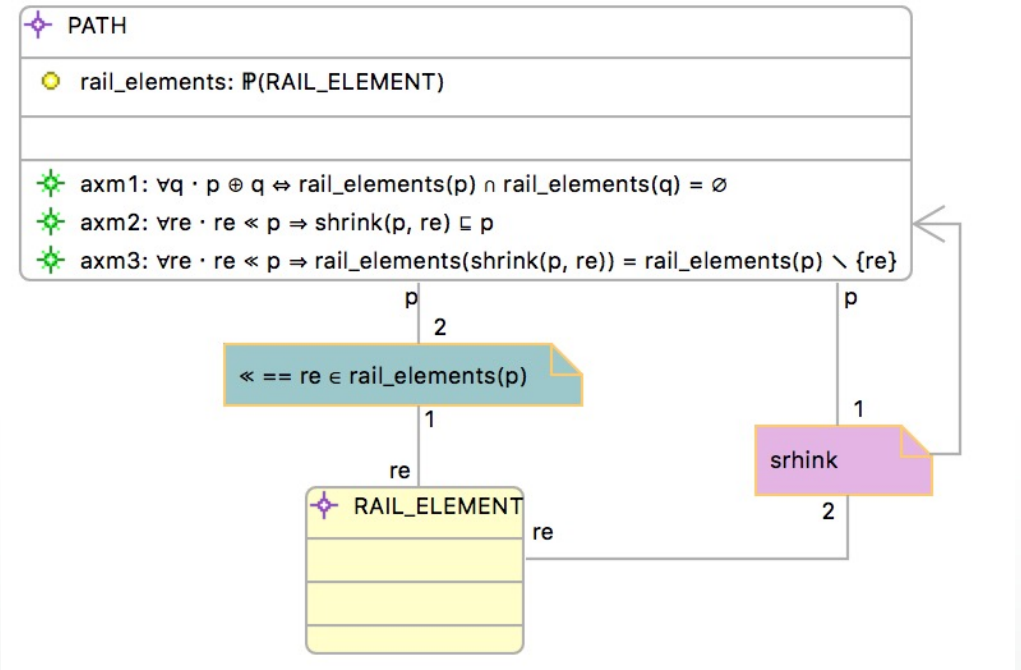


- $p \oplus q$: Paths p and q are disjoint.
- $p \sqsubseteq q$: p is a sub-path of q.

Development of Theories

Rail Elements

UNIVERSITY OF Southampton
School of Electronics and Computer Science

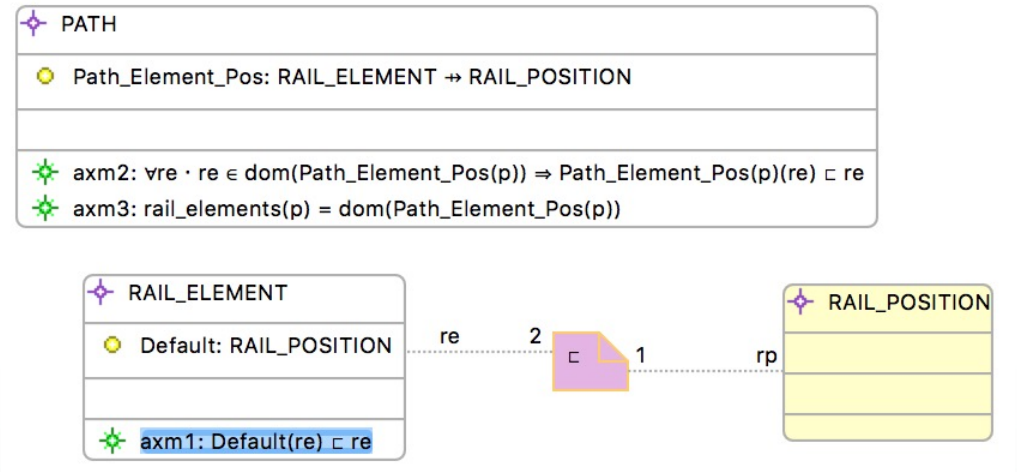


- `rail_elements(p)`: Returns the set of element of a path `p`
- `shrink(p, re)`: Remove element `re` from path `p`.
- `re ≪ p` (direct definition): Element `re` belongs to path `p`.

Development of Theories

Rail Positions

UNIVERSITY OF Southampton
School of Electronics and Computer Science

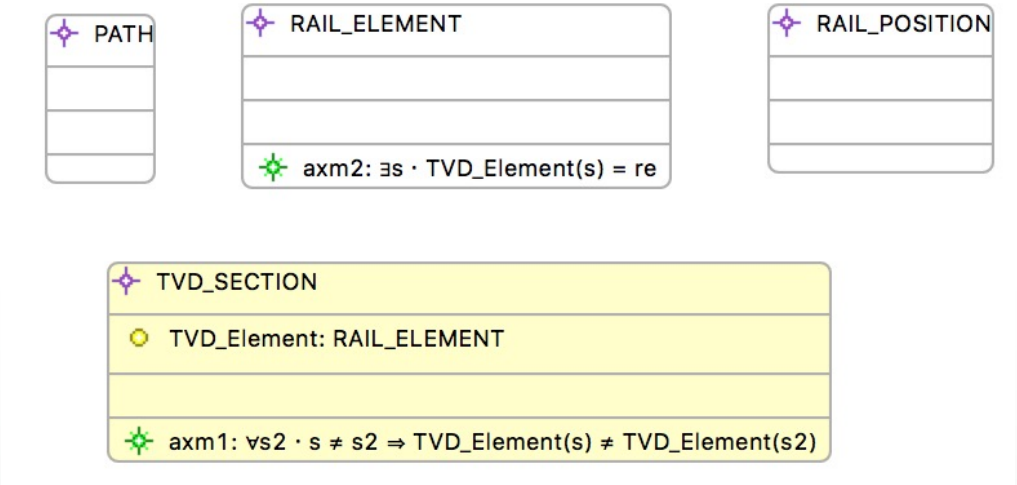


- `rp ⊏ re`: rail position `rp` is a valid for rail element `re`.
- `Default(re)`: The default position for rail element `re`
- `Path_Element_Pos(p)`: the position of the rail elements for path `p`.

Development of Theories

Vacancy Detection

UNIVERSITY OF Southampton
School of Electronics and Computer Science

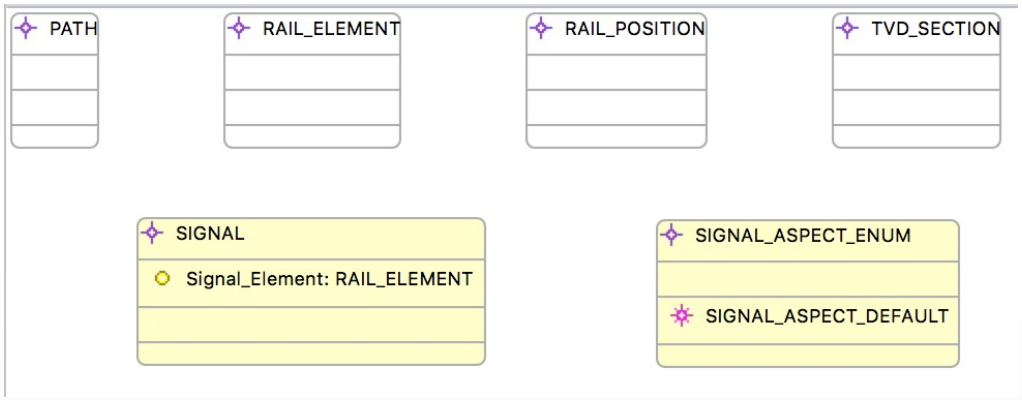


- `TVD_Element(s)`: the rail element corresponding to the TVD section `s`

Development of Theories

Signal

UNIVERSITY OF Southampton
School of Electronics and Computer Science



- `Signal_Element(s)`: the rail element that the signal `s` protects.

Summary

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Classes are linked to data types
- Attributes and associations corresponding to operators.
- Class constraints are axioms on the data types.
- Visiualsation aids the design of ADTs.
- Future work:
  - Tool support
  - Theory instantiation