LOCALES IN ISABELLE

Axiomatic Extensions Considered Harmful

→ relative consistency

As we saw already, **definitional** extension is favoured over **axiomatic** extension in Isabelle/HOL.

- Axiomatization can introduce an inconsistency.
- Example: After declaring the existence of a new type *SET* in Isabelle, it is possible to add a new axiom:

e.g.

```
axiomatization
  Member :: SET ⇒ SET ⇒ bool
where
  comprehension : ∃y.∀x. Member x y ⟷ P x
```

which enables a "proof" of the paradoxical lemma:

```
lemma member_iff_not_member : ∃y. Member y y ⟷ ¬Member y y
```

from which *False* can be derived.

- Yet, axiomatic reasoning is part of mathematics. We want to be able to carry it out safely in Isabelle.

Local axiomatic reasoning in Isabelle/HOL

Fortunately, we can reason from axioms *locally* in a sound way. For example, to prove results about groups, rings or vector spaces.

We later *instantiate* the axioms with actual groups, rings, vector spaces.

Isabelle provides a facility for doing this called **locales**.

```
locale group =
  fixes mult :: 'a ⇒ 'a ⇒ 'a  and  unit :: 'a
  assumes left_unit     : mult unit x = x
      and associativity : mult x (mult y z) = mult (mult x y) z
      and left_inverse  : ∃y. mult y x = unit
```

- In the above, *mult* and *unit* are just arbitrary names.
- For example, the integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$ form a group under the operation of addition i.e. we can instantiate mult to + and unit to 0. More on instantiation later.

Isabelle Locales

ensure safe reasoning!

- Named, encapsulated contexts, highly suitable for formalising abstract mathematics.
  - Context as a formula:

$$\bigwedge \overbrace{x_1 \ldots x_n}^{\text{parameters}} . \overbrace{[\![A_1; \ldots A_m]\!]}^{\text{assumptions}} \Longrightarrow \overbrace{C}^{\text{theorem}}$$

- Locales usually have
  - $\boxed{\text{parameters,}}$ declared using `fixes`
  - $\boxed{\text{assumptions,}}$ declared using `assumes`
- Inside a locale, definitions can be made and theorems proven based on the parameters and assumptions.
- A locale can import/extend other locales.

Locale Example: Finite Graphs

set of pairs

```
locale finitegraph =
  fixes edges :: ('a × 'a) set  and  vertices :: 'a set
  assumes finite_vertex_set : finite vertices
      and is_graph          : (u, v) ∈ edges ⟹ u ∈ vertices ∧ v ∈ vertices
begin
  inductive walk :: 'a list ⇒ bool  where
  Nil         : walk []
  |Singleton  : v ∈ vertices ⟹ walk [v]
  |Cons       : (v, w) ∈ edges ⟹ walk(w#vs) ⟹ walk (v#w#vs)
  lemma walk_edge : (v, w) ∈ edges ⟹ walk [v, w]
  ...
end
```

is it a path in the graph?

there's a path

- # is the list cons operator in Isabelle.
- The definition of this locale can be inspected by typing **thm** *finitegraph_def* in Isabelle:

finitegraph ?*edges* ?*vertices* ≡
finite ?*vertices* ∧
(∀*uv*.(*u*, *v*) ∈ ?*edges* ⟶ *u* ∈ ?*vertices* ∧ *v* ∈ ?*vertices*)

Adding Theorems to a Locale

Aside from proving a lemma within the locale definition, e.g. *walk_edge* on the previous slide, we can also state lemmas that are "in" some locale:

```
lemma (in group) associativity_bw :
    "mult (mult x y) z = mult x (mult y z)"
apply (subst associativity)
apply (rule refl)
done
```

Alternatively, we can enter a locale at the theory level using the **context keyword** and formalize new definitions and theorems:

```
context group
begin
    lemma associativity_bw :
        "mult (mult x y) z = mult x (mult y z)"
    apply (subst associativity)
    apply (rule refl)
    done
end
```

Instantiating Locales

- *Concrete* examples may be proven to be instances of a locale.
- interpretation `interpretation_name : locale_name args` generates the proof obligation that the locale predicate holds of the args.
- e.g. Example: A graph with one vertex and single edge from that vertex to itself is a concrete instance of the locale *finite_graph*.

```
interpretation singleton_finitegraph : finitegraph "{(1, 1)}" "{1}"
proof
  show "finite {1}" by simp
  next fix u v
  assume "(u, v) ∈ {(1, 1)}" then show "u ∈ {1} ∧ v ∈ {1}" by blast
qed
```

- We can prove that *singleton_finitegraph* is an instance of a finite weighted graph locale by providing a weight function as an additional argument:

```
interpretation
  singleton_finitegraph : weighted_finitegraph "{(1, 1)}" "{1}" "λ(u, v). 1"
  by (unfold_locales) simp
```

fn thr. takes a pair (u, v) and returns 1

Locale Extension

- New locales can extend existing ones by adding more parameter, assumptions and definitions. This is also known as an $\boxed{\textit{import.}}$
- The context of the imported locale i.e. all its assumptions, theorems etc. are available in the extended locale.

```
locale weighted_finitegraph = finitegraph +
  fixes weight :: ('a × 'a) ⇒ nat   (Takes an edge and returns a nat)
  assumes edges_weighted : ∀e ∈ edges.∃w. weight e = w
```

Viewed in terms of the imported *finitegraph* locale (and the weighted edges axiom), we have:

thm weighted_finitegraph_def

weighted_finitegraph ?edges ?vertices ?weight ≡
finitegraph ?edges ?vertices ∧ (∀e ∈ ?edges. ∃w. ?weight e = w)

Summary

- Axiomatization at the Isabelle theory level (i.e. as an extension of Isabelle/HOL) is not favoured as it can be unsound (see the additional exercise on the AR web page).
- Locales provide a sound way of reasoning locally about axiomatic theories.
- This was an introduction to locale declarations, extensions and interpretations.
  - There are many other features involving representation and reasoning using locales in Isabelle.
  - Reading: Tutorial on Locales and Locale Interpretation (on the AR Lecture Schedule page in Learn).