CS576 Topics in Automated Deduction

Elsa L Gunter
2112 SC, UIUC
egunter@illinois.edu
http://courses.engr.illinois.edu/cs576

Slides based in part on slides by Tobias Nipkow

March 13, 2015

Locales in Isabelle/HOL

- Locales in Isabelle introduce a theorem proving context (mathematical theory)
- Context has a collection of parameters
- And a collection of assumptions

```
locale graph =
 fixes vertices ::"'a set" and edges ::"('a × 'a)set"
 assumes IsGraph:
  "(u,v) ∈ edges ⟶ ((u ∈ vertices) ∧ (v ∈ vertices))"
```

- Defines a predicate with name of the locale, whose arguments are the locale parameters, and whose definition is the conjunction of the assumptions

```
thm graph_def
graph ?vertices ?edges ≡
  ∀ u v.  (u, v) ∈ ?edges⟶ u ∈ ?vertices ∧ v ∈ ?vertices
```


Entering a Locale Context

- Using `context` *`locale_name`* `begin` ... `end` (from top level), can enter a context where the parameters and assumptions are treated as constants and theorems
- Inside, can make definitions, prove theoerems using the parameters and assumptions of the locale

```
context graph begin

inductive reachable where
Self [intro]:  "v∈ vertices ⟹ reachable v v" |
Edge:  "⟦reachable u v; (v,w) ∈ edges⟧ ⟹ reachable u w"

lemma reachable_vertices:
assumes Reachable:  "reachable u v"
shows "u ∈ vertices ∧ v  vertices"
using Reachable
proof (rule_tac reachable.induct, assumption)
```


Instantiating a Locale

- Concrete examples may be proven to be instances of a locale
- `interpretation` *`interp_name`*`:` *`locale_name args`* generates the proof obligation that the locale predicate holds of the *`args`*
- `unfold_locale` converts locale predicate into locale assumptions

```
interpretation one:  graph "{()}" "{((),())}"
by (unfold_locales, clarsimp)
```

- Makes definitions and theorems of locale context available for the locale instance

```
term "one.reachable"
"graph.reachable {()} {((), ())}" ::  "unit ⇒ unit ⇒ bool"
thm one.reachable_vertices
graph.reachable {()} {((), ())} ?u ?v ⟹
 ?u ∈ {()} ∧ ?v ∈ {()}
```


Locale Extension

- New locales may be created from old by adding more parameters and assumptions
- All definitions and theorems of the context of the old locale and definitions and theorems of the new

```
locale labeled_graph = graph +
 fixes label ::  "'a × 'a ⇒ 'b option"
 assumes EdgesLabeled :
  "∀ e ∈ edges.  (∃ l.  (label e = Some l))"
```


Relating Existing Locales

- Locales arising in one setting may be instances of other locales from other settings
- Want to incorporate theorems and definitions from second into first item
- `sublocale` *`locale1`* ⊆ *`local2 args`* generates proof obligation that the locale predicate holds of the *`args`*

```
locale partial_order =
fixes le ::  "'a ⟹ 'a ⟹ bool" (infixl "⊑" 50)
assumes refl [intro, simp]:  "x ⊑ x"
and anti_sym [intro]:  "⟦ x ⊑ y; y ⊑ x ⟧ ⟹ x = y"
and trans :  "⟦ x ⊑ y; y ⊑ z ⟧ ⟹ x ⊑ z"

sublocale partial_order ⊆ graph "UNIV" "{(x,y)| x y.  le x y}"
by (unfold_locales, simp)
```