Interpretation of Locales in Isabelle: Theories and Proof Contexts

Clemens Ballarin

Fakultät für Informatik
Technische Universität München
85748 Garching, Germany
http://www4.in.tum.de/~ballarin

Abstract. The generic proof assistant Isabelle provides a landscape of specification contexts that is considerably richer than that of most other provers. Theories are the level of specification where object-logics are axiomatised. Isabelle's proof language Isar enables local exploration in contexts generated in the course of natural deduction proofs. Finally, locales, which may be seen as detached proof contexts, offer an intermediate level of specification geared towards reuse. All three kinds of contexts are structured, to different extents. We analyse the "topology" of Isabelle's landscape of specification contexts, by means of development graphs, in order to establish what kinds of reuse are possible.

1 Introduction

Locales are Isabelle's [12] emerging mechanism to support abstract reasoning. They are modules whose specification and theorems can be transported to other contexts by either import or interpretation. Thus specification and/or theorems can be reused.

Three kinds of contexts can be distinguished in Isabelle. These are theories, locales and proofs. Theories are the outermost level of specification, at which object-logics like HOL (Higher-Order Logic) and ZF (Zermelo-Fränkel Set Theory) are axiomatised. Locales provide a setting for specifications within these logics. Compared to theories, locales are more restricted but provide more flexible means of reuse. The third kind of contexts are the contexts of Isabelle's proof language Isar. Currently, theories are the setting of numerous formalisations, which range from abstract, like lattice theory, to concrete, like models for Java-like languages. With the facilities of locales improving — for example, packages for the definition of inductive sets and recursive functions becoming available — it is to be expected that many of these developments will move to locales in future.

Interpretation of locales in locales is the topic of [3]. There we have argued that, in order to support interactive proof, a network of import and interpretation relations need to be maintained explicitly, so that newly proved theorems can be propagated. This can be achieved with development graphs [7]. In the present paper we discuss the extension of these mechanisms to theories and proofs.

Type constructors may only be declared in theories, not locales. Interpretation of locales in theories enables to map theorems valid relative to an abstract locale specification onto concrete types of a theory. For example, the type of natural numbers can be uniformly endowed with lattice theorems, which are proved in the context of a locale for lattices, for both the lattice induced by the order relation "$\leq$" and the lattice induced by the divisibility relation "$|$". This cannot be achieved with type classes, because there the operations are fixed [16].

Interpretation in proofs is more interesting. This is a true extension of the proof language. It can raise the level of abstraction in proofs, for it enables to reason at the level of concepts rather than theorems inside a proof itself. It is also an extension of the facilities of locales since it enables to employ locales for reasoning about arbitrary *families* of structures, say groups. The inheritance mechanisms of locales themselves only enable to combine specifications in a finitary way — for example, by importing the specification of groups twice to the specification of group homomorphisms.

2 Development Graphs

Development graphs were introduced by Hutter to manage dependencies between theories in verification settings where theories are repeatedly modified, and postulated relations between theories needs to be maintained — that is, formally proved, see [7]. Development graphs can reduce the number of proof obligations caused by such changes by carefully keeping track of dependencies in the development. The following exposition of development graphs follows [7], but takes into account that in locales they are also used to propagate proved theorems. This concept was introduced in [3].

Definition 1. *A* consequence relation *is a pair* $(S, \vdash)$ *where* S *is a set of sentences and* $\vdash\ \subseteq \mathrm{Fin}(S) \times S$*, where* $\mathrm{Fin}(S)$ *denotes the set of finite subsets of* S*, is a binary relation such that*

$$\{\phi\} \vdash \phi, \qquad \text{(reflexivity)}$$

$$\Delta \vdash \phi \textit{ and } \{\phi\} \cup \Delta' \vdash \psi \textit{ implies } \Delta \cup \Delta' \vdash \psi \textit{ and} \qquad \text{(transitivity)}$$

$$\Delta \vdash \psi \textit{ implies } \{\phi\} \cup \Delta \vdash \psi. \qquad \text{(weakening)}$$

A consequence relation induces a *closure operation* on sets of sentences $\Phi \subseteq S$ defined by $\Phi^{\vdash} = \{\phi \mid \Delta \vdash \phi \text{ for some finite } \Delta \subseteq \Phi\}$. This is the set of all sentences *derivable* from Φ.

Definition 2. *A* morphism *of consequence relations from* $(S_1, \vdash_1)$ *to* $(S_2, \vdash_2)$ *is a function* $\sigma : S_1 \rightarrow S_2$ *such that* $\Delta \vdash_1 \phi$ *implies* $\sigma(\Delta) \vdash_2 \sigma(\phi)$.

Labelled directed graphs will be used to represent known dependencies between modules. A graph $G = (N, L)$ consists of nodes $n \in N$, which are module names, and links $n \xrightarrow{\sigma} m \in L$, which are labelled with consequence morphisms. There may be several links between from one node to another node, provided the morphisms are different.

As usual, reachability in graphs is defined along paths. The relation is enriched by the consequence morphism obtained from composing the labels along the path.

Definition 3. *Let $G = (N, L)$ be a labelled directed graph, where each link $n \xrightarrow{\sigma} m \in L$ is labelled with a consequence morphism σ. A node m is* reachable *from n via a consequence morphism σ, $n \xrightarrow{\sigma}_* m \in G$ for short, if $n = m$ and σ is the identity morphism* id*, or there is a $n \xrightarrow{\tau} k \in L$, and $k \xrightarrow{\rho}_* m \in G$, with $\sigma = \rho \circ \tau$.*

We will sometimes denote a graph by its set of links and write $n \xrightarrow{\sigma}_* m \in L$. The set of nodes will then be clear from the context. Likewise, we will write $n \xrightarrow{\sigma} m \in G$ instead of $n \xrightarrow{\sigma} m \in L$.

A development graph represents the import hierarchy of modules. Its links are called *definition links* and denote import relations.

Definition 4. *A* development graph *G is a finite labelled directed acyclic graph (N, L). Each node n has an associated consequence relation $(S_G(n), \vdash^n_G)$, and finite sets of sentences $\mathrm{A}_G(n), \mathrm{F}_G(n) \subseteq S_G(n)$. For each node $n \in N$, $\mathrm{A}_G(n)$ is the set of* local axioms *of n and $\mathrm{F}_G(n)$ the set of* local proved theorems *in n. Links in L are called* definition links *and are denoted $n \xrightarrow{\sigma}_{\mathrm{D}} m$. The label σ is a morphism of the consequence relations from $(S_G(n), \vdash^n_G)$ to $(S_G(m), \vdash^m_G)$.*

The sets of proved theorems are not present in Hutter's definition. They are needed to model the propagation of theorems and the control information attached to them.

The proof theoretic semantics of a development graph is given by the sentences derivable at each module node.

Definition 5. *Let G be a development graph and n be a module node. The sets of* global axioms *$\mathrm{A}^*_G(n)$ and* global proved theorems *$\mathrm{F}^*_G(n)$ of n wrt. to G are defined by*

$$\mathrm{A}^*_G(n) = \bigcup_{k \xrightarrow{\sigma}_* n \in G} \sigma(\mathrm{A}_G(k)) \qquad \textit{and} \qquad \mathrm{F}^*_G(n) = \bigcup_{k \xrightarrow{\sigma}_* n \in G} \sigma(\mathrm{F}_G(k)).$$

The theory *$\mathrm{Th}_G(n)$ of n is the set of all sentences derivable from the global axioms; $\mathrm{Th}_G(n) = \mathrm{A}^*_G(n)^{\vdash^n_G}$.*

The theory of a node depends on its local axioms and of the axioms of imported nodes. Import is transitive. Proved theorems need to be derivable. We extend the definition of development graphs and demand that $\mathrm{F}_G(n) \subseteq \mathrm{Th}_G(n)$ for all $n \in N$ in a development graph (N, L). This implies that $\mathrm{F}^*_G(n) \subseteq \mathrm{Th}_G(n)$.

Interpretation relations between modules are consequences of a development graph. They are modelled by means of *theorem links*.

Definition 6. *Let G be a development graph and n and m be nodes in G. The graph implies a* global theorem link*, denoted $G \vdash n \xrightarrow{\sigma}_{\mathrm{T}} m$, if $\mathrm{Th}_G(m) \vdash^m_G \sigma(\phi)$ for all $\phi \in \mathrm{Th}_G(n)$. It implies a* local theorem link*, denoted $G \vdash n \xrightarrow{\sigma}_{\mathrm{t}} m$, if $\mathrm{Th}_G(m) \vdash^m_G \sigma(\phi)$ for all $\phi \in \mathrm{A}_G(n)$.*

Theorem links are properties of the development graph. We will use phrases like “a theorem link has been proven” or “established” to indicate that it is implied by the development graph under consideration.

Global theorem links require the image of the entire theory of the source node to be derivable. By transitivity of the consequence relation it is sufficient if the global axioms of the source node are derivable. Local theorem links only require the local axioms to be derivable. We observe that a development graph with definition link $n \xrightarrow{\sigma}_{\mathrm{D}} m$ implies the global theorem link $n \xrightarrow{\sigma}_{\mathrm{T}} m$. This, in turn, implies the local theorem link $n \xrightarrow{\sigma}_{\mathrm{t}} m$. The following lemma, which is due to Hutter, says how a global theorem link can be decomposed into a set of local theorem links.

Lemma 1. *Let G be a development graph. Then $G \vdash n \xrightarrow{\sigma}_{\mathrm{T}} m$ if and only if $G \vdash k \xrightarrow{\sigma \circ \tau}_{\mathrm{t}} m$ for all k and τ with $k \xrightarrow{\tau}_{*} n \in G$.*

When an interpretation — that is, a global theorem link — is asserted by the user, proof obligations are generated, and it is analysed which of these follow from import or existing interpretations. The lemma enables this analysis to be at the level of links, not sentences.

3 Isabelle: Theories, Proofs and Locales

Isabelle's meta-logic is based on an intuitionistic fragment of higher-order logic with polymorphic types. The polymorphism is schematic — that is, the type system is a quantifier-free first-order language. Terms of a special type *prop* are called *propositions*. These involve connectives for universal quantification and implication: $\bigwedge x.\, \phi$ and $\phi \Longrightarrow \psi$. *Theorems* are judgements of the form

$$\{\phi_1, \dots, \phi_n\} \vdash \phi,$$

where $\phi, \phi_1, \dots, \phi_n$ are propositions. The ϕ_i are called *meta-assumptions*. We abbreviate a judgement without meta-assumptions by $\vdash \phi$. In Isabelle and some of the related literature the judgements are written in reverse notation: $\phi\ [\phi_1, \dots, \phi_n]$. Isabelle's kernel contains functions that implement the rules of a natural deduction calculus on theorems; see [14] for details. We denote the resulting derivability relation by $\Vdash$. This is a consequence relation on the set of theorems. The rules of the calculus imply that also $\vdash$ is a consequence relation, on propositions.

The difference between both relations is subtle and rooted in the nature of schematic polymorphism. Let ϕ be a proposition containing a type variable α. One may obtain $\phi[\tau/\alpha]$ by substituting a type τ for α in ϕ. While $(\vdash \phi) \Vdash (\vdash \phi[\tau/\alpha])$ holds for theorems, the direct statement on propositions $\phi \vdash \phi[\tau/\alpha]$ does not hold in general. On the other hand, the canonical map from propositions to theorems, $\iota : \phi \mapsto (\vdash \phi)$, is a consequence morphism from $\vdash$ to $\Vdash$. Well-typed substitutions are consequence endomorphisms for both $\Vdash$ and $\vdash$.

3.1 Theories

Theories encompass declarations of language, namely constants and type constructors with their syntax, and theorems over that language. Theorems are either declared or derived, and the former are axioms of the theory. Theories may import other theories. They are not parametric, and import is literal. Constants and type constructors cannot be renamed. Instead, a system of qualified names resolves name conflicts by prefixing with the theory name, which must be unique. Theorems (including axioms) in theories have no meta-assumptions.

The import hierarchy of theories is a development graph $G = (N, L)$ where N is the set of *theory nodes*. The consequence relation of each node $n \in N$ is $\Vdash$; more precisely, the subrelation on theorems of the form $\vdash \phi$. All definition links in L are embeddings.

Theorems in theories may be named, and name spaces are also relevant to these names. In addition, theorems may be decorated with attributes. These encode hints that control the automatic use of theorems. For example, the attribute [simp] makes the theorem a rewrite rule automatically used by the simplifier, [simp del] removes a theorem from the rewrite rules, and [induct set: S] declares an induction rule for the inductively defined set S.

3.2 Proofs

Isar is Isabelle's language of human-readable formal proof documents. It was developed by Wenzel [17, 18] and is radically different from tactic scripts. It is based on the principle that every object referred to in the proof text must have been introduced previously. It is not possible to refer to objects in the proof state that were generated by tactic applications, and whose meaning can only be understood by replaying the proof.

A full introduction to Isar is beyond the scope of this paper. Interested readers may consult the tutorial [11]. Here, we are only concerned with *proof contexts*. A theorem $\{\phi_1, \dots, \phi_n\} \vdash \phi$ of the meta-logic can be seen as defining a context for the proposition ϕ. This context is given by parameters $x_1, \dots, x_m$, which are the free variables occurring in the assumptions, and the assumptions $\phi_1, \dots, \phi_n$ themselves. Isar refines this idea. Its proof contexts contain, in particular, proved local propositions as additional information, and decorate assumptions by discharge rules that are applied when leaving a context. Commands for the explicit construction of contexts within a proof are provided: **fix** x introduces a new parameter, which is discharged by $\bigwedge$-introduction, and **assume** ϕ introduces a new assumption with $\Longrightarrow$-introduction as discharge rule.

The following illustrate the main building blocks of Isar texts. The resulting propositions are displayed underneath. Note that these may depend on parameters and assumptions of surrounding contexts.

```
{                              {
  fix x                          assume A
  have B x <proof>               have B <proof>
}                              }

⋀x. B x                        A ⟹ B
```

The keyword **have** introduces a local goal and is followed by a proof. It yields a local proposition — more precisely, a theorem with meta-assumptions from the context. Local propositions may be named and have attributes. It is, for example, possible to declare local rewrite rules — that is, the proof context maintains, for example, a set of rewrite rules. A $\{\cdots\}$ block can be seen as a local kind of theory. Intermediate propositions may be proved and referred to in proofs. As already indicated, blocks may be nested, and they may be used to discharge complex goals involving $\bigwedge$ and $\Longrightarrow$. In such blocks **show** is used instead of **have** for the final inner goal. Isar then ensures that the block matches the proposition of the surrounding goal.[1]

3.3 Locales

Locales were designed by Kammüller [8, 9] as a sectioning concept for tactic-based proofs. In Wenzel's reimplementation of locales for Isar [2] the focus has shifted, and there is now a stronger emphasis on locales as parametric theory modules, which are distinct from Isabelle's theories.

Locales may be seen as detached proof contexts with fixed sets of parameters and assumptions. These are identified by name. Assumptions are the module axioms. Isabelle's judgement operator $\vdash$ is a consequence relation. The set of theorems of the meta-logic is closed under substitution of types and terms for type and term variables. That is, well-typed substitutions are endomorphisms of the consequence relation. This enables to maintain locales in a structured way. A development graph $G = (N, L)$ represents the import relations between locales. (This is distinct from the graph of theories, but to simplify notation, we do not distinguish this for the moment.) The nodes $n \in N$ are *locale nodes*, and their consequence relations are $\vdash$ on the set of propositions. Locale parameters may have mixfix syntax. The list of parameters of locale n is denoted with $\mathrm{P}^*(n)$. Parameters are typed; $\mathrm{T}^*(n)$ denotes the type environment that maps the parameters of n to their types. Type variables occurring in $\mathrm{T}^*(n)$ are the *type parameters* of n.

Consistent with notation introduced in Section 2, $\mathrm{A}_G(n)$ denotes the local axioms (also called assumptions) and $\mathrm{F}_G(n)$ the local proved theorems (which are propositions) of locale n. The global assumptions are $\mathrm{A}^*_G(n)$. Interpretation relations between locales may be declared (and, of course, must be proved). A set of global theorem links T is maintained. The extended set of proved theorems of a locale

$$\mathrm{F}^*_{G,T}(n) = \bigcup_{k \xrightarrow{\sigma}_* n \in L \cup T} \sigma(\mathrm{F}_G(k)).$$

is obtained by accumulating proved theorems through any conceivable path of definition and theorem links. In [3] we have shown that this set is finite if consequence morphisms are restricted to renamings of term parameters.

4 Interpretation in Theories

Many formalisations in Isabelle are concerned with properties of objects that live in theories. In the object-logic Isabelle/HOL, in particular, many of these objects are types. Many share algebraic properties, and in order to facilitate reuse of theorems, axiomatic type classes were introduced to Isabelle [16]. These are restricted to algebraic structures with a single type parameter, and constants are shared, that is, overloaded. Locales permit more flexible forms of reuse since they may have an arbitrary number of parameters

[1] In fact, Isar is more liberal: higher-order unification takes place here.

and type parameters. In addition, locales are useful in all object-logics while axiomatic classes are not.

In order to enable reuse of locale theorems in theories, we distinguish disjoint sets of theory nodes N_{T} and locale nodes N_{L} in the development graph. Likewise, the definition links in L_{T} are between theory nodes, and the links in L_{L} between locale nodes. The set T_{L} is a set of global theorem links implied by the locale development subgraph $(N_{\mathrm{L}}, L_{\mathrm{L}})$. On the other hand, Isabelle's theory module does not support interpretations between theories, hence there are no theorem links in the theory development subgraph $(N_{\mathrm{T}}, L_{\mathrm{T}})$.

The canonical morphism $\iota : \phi \mapsto (\vdash \phi)$ enables to introduce theorem links from the locale development graph to the theory development graph. This is the set T_{T}, which is a set of local theorems links. The reason for this will become apparent in Section 4.2. The command

interpretation $l : n\ [p_1 \dots p_i]$ *<proof>*

declares a global theorem link from locale n to the current theory.[2] The arguments $p_1, \dots, p_i$ are terms over the language of the theory and determine the morphism of the theorem link: the kth parameter of n, which is the kth entry of $\mathrm{P}^*(n)$ is mapped to p_k. The substitution of type parameters is inferred. By means of Lemma 1 proof obligations are generated for local theorem links that are not implied by the development graph and existing theorem links. These are discharged by the user. Interpreted theorems are then added to the theory — but only those related to new local theorem links. This avoids unnecessary duplications. The label l is an optional theorem name prefix. It may be used to disambiguate theorems from different interpretations.

Maintaining development graph and theorem links is not only instrumental in discharging proof obligations. It also enables to propagate newly proved theorems of locales to theories that "subscribe" — that is, interpret — the locale.

4.1 Examples

A few examples are in order. The declarations below are of locales for partial orders, semi-lattices and linear orders; *partial_order* is imported by both *semi_lattice* and *linear_order*. The keyword **fixes** indicates parameters, **assumes** indicates axioms. Theorems may be added to these locales by a version of the **theorem** command.

locale partial_order =
  fixes le (**infixl** $\sqsubseteq$ 50)
  assumes refl: $x \sqsubseteq x$
    and anti_sym: $x \sqsubseteq y \wedge y \sqsubseteq x \Longrightarrow x = y$
    and trans: $x \sqsubseteq y \wedge y \sqsubseteq z \Longrightarrow x \sqsubseteq z$

locale semi_lattice = partial_order +

[2] The implementation of the command permits a *locale expression* e in place of n. This is analogous to the interpretation of locales in locales with the command **interpretation** $m \subseteq e$ *<proof>*. See [3].

fixes meet (**infixl** $\sqcap$ 70)
assumes le1: $x \sqcap y \sqsubseteq x$ **and** le2: $x \sqcap y \sqsubseteq y$
and least: $x \sqsubseteq y \wedge x \sqsubseteq z \Longrightarrow x \sqsubseteq y \sqcap z$

locale linear_order = partial_order +
assumes linear: $x \sqsubseteq y \vee y \sqsubseteq x$

Theorems of these locales may be reused in a theory *Integer* that declares a type *int* of integers. The upper half of Table 1 shows examples. The type is a semi-lattice via divisibility relation "$|$" and greatest common divisor. It is also a semi-lattice via "$\leq$" and minimum. Assume that these interpretations have been asserted and proved. Adding the interpretation that *int* is a linear order via "$\leq$" does not require to reprove that it is a partial order, since this information is stored in the set T_{T} of theorem links. Neither are the theorems of *partial_order* added to *Integer* in duplicate.

Locale	Morphism			Theory
semi_lattice	$\alpha \mapsto$ int	*le* $\mapsto |$	*meet* $\mapsto$ gcd	Integer
semi_lattice	$\alpha \mapsto$ int	*le* $\mapsto \leq$	*meet* $\mapsto$ min	Integer
linear_order	$\alpha \mapsto$ int	*le* $\mapsto \leq$		Integer
semi_lattice	$\alpha \mapsto \alpha$ set	*le* $\mapsto \subseteq$	*meet* $\mapsto \cap$	Set
semi_lattice	$\alpha \mapsto \alpha$ set	*le* $\mapsto \lambda xy.\, y \subseteq x$	*meet* $\mapsto \cup$	Set
linear_order	$\alpha \mapsto$ unit set	*le* $\mapsto \subseteq$		Set

Table 1. Some interpretations of locales in theories

4.2 Subsumption

Lemma 1 is a general characterisation of the decomposition of global theorem links into local ones with respect to a development graph. The required notion of consequence morphism — that is, substitution — is now analysed further.

The lower half of Table 1 shows interpretations of the order and lattice locales in a theory *Set*, which introduces a type constructor *set* with the usual operations. Set inclusion "$\subseteq$" and intersection "$\cap$" form a semi-lattice. So does reversed set inclusion and union "$\cup$". Standard formalisations of anti-symmetric relations in Isabelle do not provide a connective for the reversed relation. This is also the case for set inclusion. A second connective "$\supseteq$" would increase redundancy without increasing expressiveness. Since Isabelle is a higher-order framework, the reverse inclusion can be specified by $\lambda xy.\, y \subseteq x$. Application of a consequence morphism is modulo β- and η-reduction — that is, λ-terms are normalised after applying the substitution.

The last line of the table interprets set of a particular type, while the other two interpretations map the parameter type to a polymorphic set type. The type *unit* is the type with a single element. This is trivially a linear order.[3] If the previous interpretations

[3] The example is indeed not a particularly interesting one in terms of the interpretation. However, it serves well as an example of a subtype occurring in an interpretation. Examples of useful subtypes occur with axiomatic type classes.

have been established before then partial_order $\xrightarrow{\alpha \mapsto \alpha \text{ set}}_{\mathrm{t}}$ Set $\in T_{\mathrm{T}}$. This obviously subsumes partial_order $\xrightarrow{\alpha \mapsto \text{unit set}}_{\mathrm{t}}$ Set $\in T_{\mathrm{T}}$, which need not be reproved. On the other hand, the local theorem link linear_order $\xrightarrow{\alpha \mapsto \text{unit set}}_{\mathrm{t}}$ Set $\in T_{\mathrm{T}}$, is not subsumed. For this reason, global theorem links are decomposed into local ones in T_{T}.

In summary, the calculus based on Lemma 1 that infers which local theorem links of a new interpretation are implied by existing ones is over higher-order terms, and subsumption testing is required. Higher-order unification is not decidable in general, but unification of higher-order patterns, which are λ-terms where the arguments of free variables occurring in the term are η-equivalent to bound variables. See [10]. This implies that restricting consequence morphisms to substitutions that replace term variables by higher-order patterns rather than arbitrary λ-terms ensures decidability of the calculus. We have not encountered examples in practice where full λ-terms would have been required.

5 Interpretation in Proofs

Interpretation raises the level of abstraction in formal developments. It is an operation at the level of concepts (mathematical theories, or — in our case — locales) rather than theorems. The idea is also found in paper proofs and follows this pattern: an object is constructed in some context, and properties of it are proved. It may turn out that the object is an instance of groups, say. It is then common to refer to theorems from that mathematical theory in the sequel of the proof, or to apply proof techniques from that theory.

This is a form of interpretation and is used by mathematicians informally — that is, without ever referring to it as interpretation. It is desirable to add this to the proof language, and Isar's proof contexts enable to do so.

5.1 Proofs as Development Graphs

Isar proofs are built in a top-down manner by continuous refinement of goals to subgoals until eventually a goals are solved. Each subgoal is reflected explicitly as a context in the proof text. The proof is a development graph $(N_{\mathrm{P}}, L_{\mathrm{P}})$, distinct from previous developments graphs for locales and theories, whose nodes are the nested contexts. The graph is in fact a tree, since it reflects the structure of the proof. Its root represents the goal statement. Definition links point from contexts of one level to the contexts of the next level. The consequence relation of the nodes is uniformly $\vdash$. The assumptions of the goal statement are the axioms of the root node. Along each path, contexts are only lever extended, by parameters and/or assumptions. The morphisms attached to the definition links are thus embeddings. Additional assumptions introduced in a context are the local axioms of that node. The local theorems are the propositions that are proved in the context.

Consider as an example the proof of the theorem that the subgroups of a group are a complete lattice. This has been formalised in Isabelle/HOL, and is available on the author's web page at http://www4.in.tum.de/~ballarin/isabelle/SubgrpLattice.thy. We will analyse a branch of the proof tree in more detail. The statement can be formalised as follows:

$$\bigwedge G.\, \text{group}\, G \implies \text{complete\_lattice}\, (\!|\text{carrier} = \{H.\, \text{subgroup}\, H\, G\}, \sqsubseteq = \text{subgrp}\, G|\!) \tag{1}$$

Thick parentheses $(\!|\cdots|\!)$ denote records. The lattice $(\!|\text{carrier} = \{H.\, \text{subgroup}\, H\, G\}, \sqsubseteq = \text{subgrp}\, G|\!)$ will be abbreviated by L in the sequel.

The context of this statement is given by parameter G and assumption group G. One proof of the statement involves showing the existence of a greatest lower bound (infimum) of each non-empty subset A of the carrier $\{H.\, \text{subgroup}\, H\, G\}$. The subgoal is:

$$\bigwedge A.\, A \subseteq \text{carrier}\, L \wedge A \neq \{\} \implies \exists I.\, \text{greatest}\, L\, I\, (\text{Lower}\, L\, A) \tag{2}$$

Note that this is relative to the context of the previous goal. The proof of this goal requires, amongst others, that the intersection of the groups in A is a subgroup of each $H \in A$:

$$\bigwedge H.\, H \in A \implies \bigcap A \sqsubseteq_L H \tag{3}$$

The path of the development graph corresponding to Subgoals (1) to (3) is depicted in Figure 1. To the right of the nodes, assumptions of the goals are displayed. In development graph terminology these are local axioms. Note that by Definition 5 they have to be read incrementally.

$$\begin{tikzcd}
\text{submagma} \arrow[d] & & (1) \arrow[d] & \text{group}\, G \\
\text{submonoid} \arrow[d] & & (2) \arrow[d] & A \subseteq \text{carrier}\, L \wedge A \neq \{\} \\
\text{subgroup} \arrow[urr, dashed, bend left, "\begin{array}{c} H \mapsto \bigcap A \\ G \mapsto G \end{array}"] \arrow[rr, dashed, bend right, "\begin{array}{c} H \mapsto H \\ G \mapsto G \end{array}"'] & & (3) & H \in A
\end{tikzcd}$$

Fig. 1. Locale hierarchy and a proof that interprets subgroup locales.

Completing the proof of (3) requires to show that $\bigcap A$ is a subgroup of H in G. This follows, because both $\bigcap A$ and H are subgroups of G. An Isar proof of this is reproduced in Figure 2(a). It is required to show that the set $\bigcap A$ is closed with respect to group operations of G. The proof is complicated considerably, because the notion of H being a subgroup of G is specified incrementally through the hierarchy of locales shown in Figure 1 on the left. It is a realistic assumption that an algebraic library is structured in that way.

5.2 Theorem Links

Interpretation enables to abstract away from these library structure details in the proof text. The command

interpret $l : n\ [p_1 \dots p_i]$ *<proof>*

is analogous to the corresponding command for interpretation in theories, but is available in proofs.[4] It maintains a set of local theorem links T_{P} from the locale development subgraph to the proof development subgraph. Links are persistent in the scope of the proof only. They facilitate reuse of interpretations, for example if a hierarchy of locales is interpreted incrementally.

```
show ⋂A ⊑_L H
  proof (simp, rule_tac subgrpI)
    show H ⊆ carrier G
      by (rule submagma.subset [OF subgroup.is_submagma, OF subgroupH])
  next
    show ⋀xy. x ∈ ⋂A ∧ y ∈ ⋂A ⟹ x ⊗ y ∈ ⋂A
      by (rule submagma.m_closed [OF subgroup.is_submagma, OF Int_subgroup])
  next
    show 1 ∈ ⋂A
      by (rule submonoid.one_closed [OF subgroup.is_submonoid, OF Int_subgroup])
  next
    show ⋀x. x ∈ ⋂A ⟹ inv x ∈ ⋂A
      by (rule subgroup.m_inv_closed [OF Int_subgroup])
  qed
```

(a) Proof without interpretation.

```
show ⋂A ⊑_L H
  proof (simp, rule_tac subgrpI)
    show H ⊆ carrier G by (rule H.subset)
  next
    show ⋀xy. x ∈ ⋂A ∧ y ∈ ⋂A ⟹ x ⊗ y ∈ ⋂A by (rule Int.m_closed)
  next
    show 1 ∈ ⋂A by (rule Int.one_closed)
  next
    show ⋀x. x ∈ ⋂A ⟹ inv x ∈ ⋂A by (rule Int.m_inv_closed)
  qed
```

(b) Proof with interpretation.

```
show ⋂A ⊑_L H by (auto intro: subgrpI)
```

(c) Interpretation makes a more concise proof possible.

Fig. 2. Comparison of proofs without and with interpretation.

In the proof described in the previous section, interpretation can be applied fruitfully twice.

In Proof Context (2): **interpret** Int: subgroup [$\bigcap A\ G$] *<proof>*
In Proof Context (3): **interpret** H: subgroup [$H\ G$] *<proof>*

[4] The name is different because this is required, for technical reasons, by the front end of the proof assistant.

The result is a simpler proof of Goal (3), which is shown in Figure 2(b). This is much cleaner, because manual interpretations of theorems by means of "[OF ...]" are no longer necessary. If, finally, the corresponding locale theorems were declared as rewrite rules with the attribute [simp] in the locale, the proof can be collapsed to a single line. See Figure 2(c).

6 Conclusion

The facilities for the interpretation of locales in theories and proofs described here have been implemented in Isabelle and are available with the release of Isabelle 2005. More information on the commands **interpretation** and **interpret** can be found in the corresponding Isar Reference Manual [15].

Interpretation is implemented in a hand full of provers, namely IMPS [6], PVS [13] and Coq [5], and development graphs are implemented in the development graph manager Maya [1]. In these systems only a single notion of context prevails, namely that of a theory. A comparison of the facilities of these systems and locales was given in our paper on interpretation of locales in locales [3].

The present work goes beyond that in that interpretation is extended to other contexts. Development graphs provide a framework making it possible to describe all three kinds of contexts available in Isabelle, namely theories, proofs and locales, in a uniform way. The result is a development graph where theory nodes, proof nodes and locale nodes are distinguished, and where theorem links, which represent interpretations, may only start from locale nodes but may point to any kind of node.

The distinction of theories and locales is rooted both in Isabelle being a framework for the specification of logics, and in types not being first-class citizens of higher-order logic. Interpretation of type constructors, which can be declared in theories but not in locales, would either require to apply context morphisms to proof objects, or to give up the concept of an LCF-style kernel. Given this distinction, the interpretation of locales in theories is useful. Maintaining theorem links explicitly makes it possible for theories to subscribe to locales, so that theorems become available there, whenever added to locales.

Interpretation in proof contexts is most exciting in combination with human-readable proofs. It enables proof writers to mimic a technique common in mathematical texts, namely to insert theorems from different mathematical theories in a proof as required. We have implemented interpretation in Isar proofs. Our experiment, a proof that the subgroups of a group form a complete lattice, shows that this can make proofs considerably more concise. The experiment also shows how locales can be applied to a family of objects, here the family of subgroups of a group, within a proof, while structured specifications cannot deal with that directly.

Consequence morphisms may map parameters to higher-order terms. By restricting these to higher-order patterns, the problem whether new interpretations are implied by existing ones can be shown to be decidable for interpretations in theories and proofs.

References

1. S. Autexier, D. Hutter, T. Mossakowski, and A. Schairer. The development graph manager Maya. In H. Kirchner and C. Ringeissen, editors, *Algebraic Methodology and Software Technology, AMAST 2002, Saint-Gilles-les-Bains, Reunion Island, France*, LNCS 2422, pages 495–501. Springer, 2002.
2. C. Ballarin. Locales and locale expressions in Isabelle/Isar. In Berardi et al. [4], pages 34–50.
3. C. Ballarin. Interpretation of locales in Isabelle: Managing dependencies between locales. Technical Report TUM-I0607, Technische Universität München, 2006.
4. S. Berardi, M. Coppo, and F. Damiani, editors. *Types for Proofs and Programs, TYPES 2003, Torino, Italy*, LNCS 3085. Springer, 2004.
5. J. Chrzaszcz. Modules in Coq are and will be correct. In Berardi et al. [4], pages 130–136.
6. W. M. Farmer, J. D. Guttman, and F. J. Thayer. Little theories. In D. Kapur, editor, *Automated deduction, CADE-11: Saratoga Springs, NY, USA*, LNCS 607, pages 567–581. Springer-Verlag, 1992.
7. D. Hutter. Management of change in structured verification. In *Automated Software Engineering, ASE 2000, Grenoble, France*, pages 23–31. IEEE Computer Society, 2000.
8. F. Kammüller. *Modular Reasoning in Isabelle*. PhD thesis, University of Cambridge, Computer Laboratory, Aug. 1999. Also Technical Report No. 470.
9. F. Kammüller, M. Wenzel, and L. C. Paulson. Locales: A sectioning concept for Isabelle. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Théry, editors, *Theorem Proving in Higher Order Logics: TPHOLs'99, Nice, France*, LNCS 1690, pages 149–165. Springer, 1999.
10. T. Nipkow. Functional unification of higher-order patterns. In *Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 64–74, 1993.
11. T. Nipkow. Structured proofs in Isar/HOL. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs (TYPES 2002)*, LNCS 2646, pages 259–278. Springer, 2003.
12. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. LNCS 2283. Springer, 2002.
13. S. Owre and N. Shankar. Theory interpretation in PVS. Technical Report CSL-01-01, SRI, Apr. 2001.
14. L. C. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5(3):363–397, 1989.
15. M. Wenzel. The Isabelle/Isar reference manual. Part of the Isabelle distribution, available at http://isabelle.in.tum.de.
16. M. Wenzel. Type classes and overloading in higher-order logic. In E. L. Gunter and A. Felty, editors, *Theorem proving in higher order logics: 10th International Conference, TPHOLs '97, Murray Hill, NJ, USA, August 19–22, 1997: proceedings*, LNCS 1275, pages 307–322. Springer, 1997.
17. M. Wenzel. *Isabelle/Isar — a versatile environment for human-readable formal proof documents*. PhD thesis, Technische Universität München, 2002. Internet publication, http://tumb1.biblio.tu-muenchen.de/publ/diss/in/2002/wenzel.html.
18. M. Wenzel. Structured induction proofs in Isabelle/Isar. MKM, 2006.