Interpretation of Locales in Isabelle: Managing Dependencies between Locales

Clemens Ballarin

Fakultät für Informatik
Technische Universität München
85748 Garching, Germany
ballarin@in.tum.de

Abstract

Locales are the theory development modules of the Isabelle proof assistant. Interpretation is a powerful technique of theorem reuse which facilitates their automatic transport to other contexts. This paper is concerned with the interpretation of locales in the context of other locales. Our main concern is to make interpretation an effective tool in an interactive proof environment. Interpretation dependencies between locales are maintained explicitly, by means of a development graph, so that theorems proved in one locale can be propagated to other locales that interpret it. Proof tools in Isabelle are controlled by sets of default theorems they use. These sets are required to be finite, but can become infinite in the presence of arbitrary interpretations. We show that finiteness can be maintained.

1 Introduction

The need for modularisation in formal development has been recognised early, and has been a major line of research in the algebraic specification community. Modularisation is also important in provers, and most proof assistants have a notion of theory development module (usually these modules are called theories). Only a small number of these systems support *theory interpretation*, which is a powerful technique of theorem reuse. These are IMPS [6], PVS [12] and Coq [5].

The present paper is concerned with the implementation of theory interpretation for locales, which are the theory development modules of the proof assistant Isabelle [11]. This integration is based upon three principles, which are derived from the practice of interactive proof and are aimed at providing effective support for the user.

Modules can either import or interpret modules. While import adds specifications and theorems of one module to another, interpretation only adds theorems, but specifications have to be discharged. Import and interpretation is along morphisms

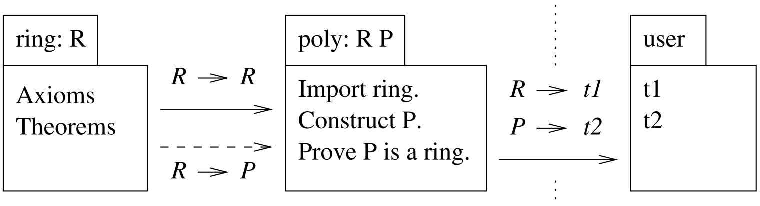

Figure 1: Chain of modules with import and interpretation.

that map sentences to sentences. *It is necessary to manage these dependencies,* so that theorems can be propagated between modules.

Modules are not just collections of theorems (together with declarations of language and specification, of course), but they contain information on the set-up of proof tools as well. Providing control information is part of the theory developer's work! Typically, a collection of theorems is identified that works well with a particular proof procedure. In Isabelle, this information is attached to theorems using attributes. *Control information should be propagated automatically.* This implies the automatic propagation of collections of theorems.

Interpretations have to be shown valid by discharging specifications in the context of the module where the interpretation takes place. *Some proof obligations may already follow from existing import or interpretation relations.* These should be solved by the system, not presented to the user.

The need for the first design goal arises in a scenario as illustrated in Figure 1. A library module *poly* of polynomials P over a coefficient ring R is imported in a module *user*. The polynomial module itself imports ring specification and theorems for the coefficient ring, and interprets them for the polynomials. If now, in *user*, a ring theorem is required for both $t1$ and $t2$ that is not available in the library, this can, of course be proved locally, but it is necessary to prove it twice. Instead, it is desirable to add it to *ring* and propagate it to the other modules.

What is addressed by the last goal is a simple form of change management, and can be achieved with development graphs. The second design goal poses a problem. While the import hierarchy of modules is acyclic, interpretation morphisms can introduce cycles. Then the propagation of theorems need no longer be a finite process. There are examples of desirable module hierarchies where this is the case. It turns out that a restriction to morphisms between locales in the current design prevents these cycles from generating infinite contexts.

Users of Isabelle may be aware that interpretation of locales is not only possible in the context of locales, but also in the context of Isabelle theories and in proof contexts. In this paper we only address the interpretation of locales in locales.

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 need 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. A number of operations is supported, including changes in the import hierarchy of theories and the addition and deletion of axioms. We only require few of these facilities. The following exposition of development graphs follows [7], but is simplified.

Definition 1 *A* consequence relation *is a pair* $(S, \vdash)$ *where* S *is a set of sentences and* $\vdash \subseteq \mathrm{Fin}(S) \times S$ *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 a consequence relation* $(S, \vdash)$ *is a function* $\sigma : S \to S$ *such that* $\Delta \vdash \phi$ *implies* $\sigma(\Delta) \vdash \sigma(\phi)$.

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 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 $n \xrightarrow{\sigma} m \in L$.

Definition 4 *A* development graph G *is a finite labelled directed acyclic graph* (N, L) *with two functions* $\mathrm{A}_G, \mathrm{F}_G : N \to \mathrm{Fin}(S)$. *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. Each link in L is labelled with a morphism $\sigma : S \to S$ of the consequence relation $\vdash$. Links are called* definition links *and are denoted $n \xrightarrow{\sigma}_{\mathrm{D}} m$.*

A development graph represents the import hierarchy of modules. Its links denote import relations. Hutter demands a consequence relation for each node of the development graph. We only require a single, global one. This will be the consequence relation given by Isabelle's meta-logic. 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*

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

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

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* $\mathrm{Th}_G(m) \vdash \sigma(\phi)$ *for all $\phi \in \mathrm{Th}_G(n)$. It implies a* local theorem link*, denoted $G \vdash n \xrightarrow{\sigma}_{\mathrm{t}} m$, if* $\mathrm{Th}_G(m) \vdash \sigma(\phi)$ *for all $\phi \in \mathrm{A}_G(n)$.*

Theorem links are properties of the development graph. We will uses phrases like "a theorem link has been proved" 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. It is the key to adding interpretations to locales.

3 Locales and Locale Expressions

Isabelle's meta-logic [13] is based on an intuitionistic fragment of simply typed λ-calculus. The type system is extended to a first-order language providing type variables and hence (schematic) polymorphism. Isabelle is implemented following the LCF-approach. A relatively small kernel implements the deductive system. Theorems can only be derived using functions of the kernel, which implement the inference rules. The meta-logic uses natural deduction. Theorems depend on meta-level assumptions. The system works with assertions of the form

$$\phi \quad [\phi_1, \ldots, \phi_n],$$

where $\phi_1, \ldots, \phi_n$ are the assumptions. This can also be read as $\{\phi_1, \ldots, \phi_n\} \vdash \phi$, and the rules of the calculus imply that $\vdash$ 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. Hence, substitutions are morphisms of the consequence relation.

Locales, which were designed by Kammüller [8, 9] as a module system for Isabelle, are based on this observation.[1] A locale is defined by a set of axioms $\phi_1, \ldots, \phi_n$ (called *assumptions* in the locale jargon) and contains theorems of the form $\phi \quad [\phi_1, \ldots, \phi_n]$. Parameters — the variables that may be substituted by morphisms — are managed explicitly. Only term parameters have to be declared, though. Type parameters can be inferred, because the type system admits principal types to be inferred. In the sequel, when referring to parameters of a locale, we mean its term parameters, unless stated otherwise.

Locale expressions were introduced by Wenzel; see [3]. As is common in other specification languages, they provide means for constructing complex locales from simpler ones. Locales are named, and a locale name is a locale expression. For locale expressions e, e_1 and e_2, $e_1 + e_2$ denotes the combined locale, called *merge*, and $e\ q_1 \ldots q_n$ is a *renaming*. The latter denotes the locale obtained by renaming parameters of e, where parameters are referred to by their canonical order, which is defined in the following paragraph. Merge simply computes the unions of the axioms and theorems of the merged locales, respectively. Parameters are identified

[1] In contrast, modules of other provers are usually based on the derivability relation of the calculus, which is also a consequence relation.

by name and are shared in the merged locale.[2] A parameter may have different types in the locales that are merged. The type in the merged locale is the most general unifier. If a unifier does not exist, the locales are not compatible, and constructing the expression fails.

In the remainder of this section, the exposition of locales [3] is summarised and related to development graphs. Additionally, type inference of locale parameters is briefly touched upon. This was not covered in [3].

3.1 Parameters, Import and Theorems

A locale declaration consists, besides of the locale name, of an imported locale expression and of declarations of local parameters and local axioms (also called assumptions). The declarations are the *body specification* of the locale. Both import and body may be empty. The body extends the imported expression. The development graph is extended by information on the parameters of its nodes. For locale n, $\mathrm{P}(n)$ is the list of local parameters. These are the parameters specified in the body. The list of parameters $\mathrm{P}^*(e)$ of a locale expression, and, as a special case, all parameters of a locale, is defined by

$$\begin{aligned}
\mathrm{P}^*(n) &= \mathrm{P}^*(\mathrm{I}(n))\,\overline{@}\,\mathrm{P}(n) \\
&\quad \text{where } n \text{ is a locale name} \\
\mathrm{P}^*(e\,q_1\dots q_n) &= [q_1,\dots,q_n,p_{n+1},\dots,p_m], \\
&\quad \text{where } \mathrm{P}^*(e) = [p_1,\dots,p_m] \\
&\quad \text{and } q_1,\dots,q_n,p_{n+1},\dots,p_m \text{ are all distinct} \\
\mathrm{P}^*(e_1+e_2) &= \mathrm{P}^*(e_1)\,\overline{@}\,\mathrm{P}^*(e_2)
\end{aligned}$$

The operation $\overline{@}$ concatenates two lists, omitting elements from the second list that are also present in the first; $\mathrm{I}(n)$ denotes the import of locale n. The order of parameters in $\mathrm{P}^*(n)$ is the *canonical order* of parameters of n.

Parameter types may be declared, or are inferred from the axioms. A *type environment* is a function that maps parameter names to their types. Let $\mathrm{T}(n)$ denote the environment derived from the body specification of locale n. It determines the types of the local parameters. It may also type imported parameters, if they occur in the axioms. The type environment of a locale expression, and a locale including import, is obtained by unifying parameter types.

$$\begin{aligned}
\mathrm{T}^*(n) &= \mathrm{T}^*(\mathrm{I}(n))\,\tilde{\cup}\,\mathrm{T}(n) \\
\mathrm{T}^*(e\,q_1\dots q_n) &= \mathrm{T}^*(e)\circ[q_1,\dots,q_n/p_1,\dots,p_n], \\
&\quad \text{where } \mathrm{P}^*(e) = [p_1,\dots,p_m],\ n \le m \\
\mathrm{T}^*(e_1+e_2) &= \mathrm{T}^*(e_1)\,\tilde{\cup}\,\mathrm{T}^*(e_2)
\end{aligned}$$

[2] In the current implementation of locales, sharing is simulated through normalisation of locale expressions. Hence parameters must be imported using the predeclared locale var, rather than fixed as we have done in the examples to follow. See [3, Section 4.3].

Unification of two environments μ and ν, denoted by $\mu \tilde{\cup} \nu$, is obtained by first renaming type variables in ν to make them distinct from μ, obtaining ν', and then computing the simultaneous most general unifier σ of $\sigma(\mu x) = \sigma(\nu' x)$ for all parameters x occuring in the domain of both μ and ν'. The unified typing is $\mu \tilde{\cup} \nu = \sigma \circ \mu \cup \sigma \circ \nu'$. The *type parameters* of a locale are the type variables occurring in its type environment.

An import hierarchy of locales can be expressed as a development graph G. Its nodes are locale names. The local axioms of locale n are $\mathrm{A}_G(n)$. The set of local proved theorems $\mathrm{F}_G(n)$ is initially equal to $\mathrm{A}_G(n)$. Import leads to definition links. Let n be a locale importing expression e. For each locale name m occuring in e, the graph contains a definition link $m \xrightarrow{\sigma}_{\mathrm{D}} n$. The consequence morphism σ is a substitution that maps type parameters of m to types in n and parameters of m to parameters of n. The expression e determines an injective renaming of the parameters $\mathrm{P}^*(m)$ of m. This determines the effect of σ on term parameters. Types of parameters in n are instances of the types of the corresponding parameters in m. The effect of σ is given by simultaneously matching these instance pairs.

Locales support proving in and adding theorems to locales. This is done in the following way. Assume that a theorem ψ is to be added to locale n. The sets of global axioms $\mathrm{A}^*_G(n)$ and global theorems $\mathrm{F}^*_G(n)$ provide a context in which the proof of ψ may be constructed. Not only does this context provide the known theorems. In Isabelle it also initialises proof tools with default sets of theorems. These default sets may be modelled with sets of proved theorems. The resulting theorem is added to $\mathrm{F}_G(n)$ and available in future proofs, both in the context of locale n or in contexts importing n.[3]

3.2 Examples

Before considering examples of locale declarations, which will be from group theory, we introduce a number of objects. These are in the Isabelle theory that will also contain the locales.

typedecl α group
consts prod :: α group $\Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha$ (**infixl** $\cdot_1$ 70)
  one :: α group $\Rightarrow \alpha$ (1_1)
  inv :: α group $\Rightarrow \alpha \Rightarrow \alpha$ (inv$_1$ _ [81] 80)

The first declaration is of a type constructor representing groups. Its argument is the carrier type of the group. The type appears in the first argument of each of the constants, which in turn represent group operations; $\Rightarrow$ is the constructor of function types. Type and constants are not further specified. Syntax annotations admit to write $x \cdot_G y$ for prod $G\, x\, y$, 1_G for one G and inv$_G\, x$ instead of inv $G\, x$, with the usual precedences. The first locale specifies semigroups.

[3] A theorem $\psi \in \mathrm{F}_G(n)$ depends on axioms of n. The theorem stored by the locale implementation is an assertion $\psi \quad [\phi_1, \ldots, \phi_n]$, where all assumptions $\phi_i \in \mathrm{A}^*_G(n)$.

```
locale semigroup =
  fixes $G$
  assumes assoc: $(x \cdot_G y) \cdot_G z = x \cdot_G (y \cdot_G z)$
```

The keywords **fixes** and **assumes** are followed by parameters and assumptions, respectively. The assumption is named, and the name can be used to reference the corresponding theorem. The type of the parameter is inferred from the assumption. It is α group, hence α is the type parameter of the locale. Term variables in assumptions that are not parameters are implicitly universally quantified. The product operation could have equally well been chosen as the parameter, as is the case in the examples of [3]. Here we will consider extending the locale to a specification of groups, and it will be more convenient to have a single parameter per group, rather than a parameter per operation.[4]

Semigroups may be extended in various ways to monoids and groups.

```
locale monoid = semigroup +
  assumes lone [simp]: $1_G \cdot_G x = x$ and rone [simp]: $x \cdot_G 1_G = x$
locale lgroup = semigroup +
  assumes lone [simp]: $1_G \cdot_G x = x$ and linv [simp]: $\mathrm{inv}_G\, x \cdot_G x = 1_G$
locale rgroup = semigroup +
  assumes rone [simp]: $x \cdot_G 1_G = x$ and rinv [simp]: $x \cdot_G \mathrm{inv}_G\, x = 1_G$
```

Assumptions in these declarations have the attribute simp. Its effect is that the theorems corresponding to the assumptions are added to the default set of rewrite rules when creating a proof context for a new theorem. The next example illustrates this.

```
theorem (in lgroup) lcancel: $x \cdot_G y = x \cdot_G z \leftrightarrow y = z$
proof
  assume $x \cdot_G y = x \cdot_G z$
  then have $(\mathrm{inv}_G\, x \cdot_G x) \cdot_G y = (\mathrm{inv}_G\, x \cdot_G x) \cdot_G z$ by (simp only: assoc)
  then show $y = z$ by simp
qed simp
```

A cancellation law is added to the local proved theorems of lgroup. The proof is in the Isar proof language. Explaining Isar in detail is beyond the scope of this paper; interested readers may consult the tutorial [10]. It is sufficient to note that the proof uses Isabelle's rewrite method simp. The code inside the proof-qed-block is concerned with the direction from left to right. In the first invocation of simp, the theorem assoc, which is inherited from semigroup, is supplied as a rewrite rule. The second use of simp is with the default set. The theorems lone and linv are used in particular. The last invocation of simp deals with the direction from right to left.

The specification of homomorphisms between groups is our final example.

[4] The type and constants simulate a record type. Record types are available in Isabelle/HOL [11, Chapter 8.3], and can be used for the same purpose.

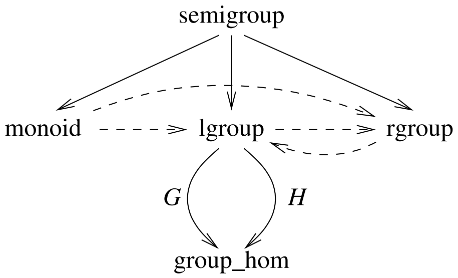

Figure 2: Development graph and implied global theorem links of group locales.

> **locale** group_hom = lgroup G + lgroup H +
> **fixes** h
> **assumes** $h\,(x \cdot_G y) = h\,x \cdot_H h\,y$

Two copies of the group locale are imported. Type inference assigns α group to G and β group to H. The type of parameter h is $\alpha \Rightarrow \beta$. The set of global proved theorems of group_hom contains two copies of each global proved theorem of lgroup, one for the parameter G and one for H. Theorem names are disambiguated by suitable prefixes. For details, see [3].

The import graph for the locales is shown in Figure 2. Definition links are continuous arrows, and consequence morphisms are indicated by renaming.

3.3 Identifiers

The sets $\mathrm{A}^*_G(n)$ and $\mathrm{F}^*_G(n)$ of a locale n are not computed directly. An intermediate, symbolic representation of locales is used. This is based on *identifiers*. An identifier is a pair of a locale name and a list of parameters. An identifier represents the body specification of a locale and its local proved theorems. The list of parameters specifies a renaming of the parameters in body specification and local theorems. We write identifiers using the syntax of renamings. The identifier $n\;q_1 \ldots q_m$ represents the renamed body of locale n. Unlike with locale expressions, for an identifier it is required that all parameter names are specified — that is, m is the length of $\mathrm{P}^*(n)$.

A locale expression e is *flattened* to a list of identifiers $\mathrm{N}(e)$ by the following algorithm.

$$\begin{aligned}
\mathrm{N}(n) &= \mathrm{N}(\mathrm{I}(n))\,\overline{@}[(n, \mathrm{P}^*(n))] \\
\mathrm{N}(e\;q_1 \ldots q_n) &= \mathrm{N}(e)[q_1, \ldots, q_n/p_1, \ldots, p_n], \\
&\quad \text{where } \mathrm{P}^*(e) = [p_1, \ldots, p_m],\; n \le m \\
\mathrm{N}(e_1 + e_2) &= \mathrm{N}(e_1)\,\overline{@}\,\mathrm{N}(e_2)
\end{aligned}$$

For example, flattening the locales from the previous paragraph yields

$$\begin{aligned}
\mathrm{N}(\text{semigroup}) &= [\text{semigroup}\,G] \\
\mathrm{N}(\text{lgroup}) &= [\text{semigroup}\,G, \text{lgroup}G] \\
\mathrm{N}(\text{group\_hom}) &= [\text{semigroup}\,G, \text{lgroup}\,G, \text{semigroup}\,H, \text{lgroup}\,H, \\
&\quad\ \text{group\_hom}\,G\,H\,h]
\end{aligned}$$

These lists can be used to obtain the axioms and proved theorems of a locale.

Lemma 2 *Let G be the development graph constructed from* P*,* T *and* I*. Then*

$$\begin{aligned}
\mathrm{A}^*_G(n) &= \bigcup_{m\,q_1\ldots q_i \in \mathrm{N}(n)} \tau_{m\,q_1\ldots q_i}(\mathrm{A}_G(m)), \\
\mathrm{F}^*_G(n) &= \bigcup_{m\,q_1\ldots q_i \in \mathrm{N}(n)} \tau_{m\,q_1\ldots q_i}(\mathrm{F}_G(m)),
\end{aligned}$$

where $\tau_{m\,q_1\ldots q_i}$ is the substitution with term part $[q_1, \ldots, q_i]/P^*(m)$. The types of parameters of n are instances of the types of the corresponding parameters of m. The simultaneous matcher of these instance pairs gives the type part of the substitution.

Proof. The argument is, of course, the same for A^*_G and F^*_G. The set of global axioms of n, $\mathrm{A}^*_G(n)$ is defined as the union of all $\sigma(A_G(k))$ with $k \xrightarrow{\sigma}_* n \in G$. That is, we have to show a one-to-one correspondence between pairs of nodes and consequence morphisms on one side, and identifiers on the other side.

We define a map f from identifiers in $\mathrm{N}(n)$ to pairs of nodes and consequence morphisms by

$$f(k\,q_1\ldots q_i) = (k, \tau_{m\,q_1\ldots q_i}).$$

Let $k\,p_1\ldots p_i$ and $l\,q_1\ldots q_j$ be different identifiers. If $k = l$ then $i = j$, and $\tau_{k\,p_1\ldots p_i}$ and $\tau_{k\,q_1\ldots q_i}$ differ on at least one global parameter of k. Otherwise $k \neq l$. In both cases $f(k\,p_1\ldots p_i) \neq f(l\,q_1\ldots q_j)$. Hence f is injective.

On the other hand, if $k \xrightarrow{\sigma}_* n \in G$, there is a path

$$k = n_0 \xrightarrow{\sigma_1}_{\mathrm{D}} \cdots \xrightarrow{\sigma_t}_{\mathrm{D}} n_t = n$$

in G. Each n_{j+1} imports n_j under a consequence morphism σ_{j+1}, which renames the global parameters of n_j. By construction of G, this renaming is defined by $\mathrm{I}(n_{j+1})$, the imported locale expression of n_{j+1}. By induction on j, $k\,q_1\ldots q_i \in \mathrm{N}(n_{j+1})$, where $q_1, \ldots, q_i$ are the images of the global parameters of k under $\sigma_{j+1} \circ \ldots \circ \sigma_1$. It follows that $k\,q_1\ldots q_i \in \mathrm{N}(n)$, where $q_1, \ldots, q_i$ are the images of the parameters of k under σ. Hence f is surjective. This completes the proof. □

4 Interpretation

Let us assume that theorems have been proved for the locales in Section 3. For example, the law of left cancellation $x \cdot_G y = x \cdot_G z \leftrightarrow y = z$ follows easily in lgroup and the law of right cancellation $y \cdot_G x = z \cdot_G x \leftrightarrow y = z$ in rgroup. Groups are monoids and the specifications of lgroup and rgroup are equivalent — see, for example, van der Waerden's textbook on Algebra [14]. Since lgroup and rgroup are equivalent, it is desirable to have all theorems of rgroup available in proofs in lgroup and vice versa. This can be achieved with theory interpretation. Once it has been proved that the assumptions of one locale imply the assumptions of another locale (under a consequence morphism), the proved theorems of the latter (under the same consequence morphism) can be included in the proved theorems of the former.

4.1 Adding Interpretation to Locales

Our integration of interpretation with locales provides the new command

interpretation $m \subseteq e \quad \varphi$

for declaring interpretation dependencies between locales. The command declares that the global axioms of the named locale m imply the global axioms of the locale expression e. Proof obligations are computed, which must be discharged. Hence a proof script φ follows. Dependencies are maintained by means of a development graph extended by a set of global theorem links. Only interpretation dependencies that are new lead to proof obligations. Theorem links change the set of global proved theorems of a locale node. The set of global proved theorems of a locale m is provided whenever theorems of that locale are to be proved. This occurs in two situations: when theorems are added to $\mathrm{F}_G(m)$ by means of the **theorem** command, and when new theorem links are proved with the **interpretation** command.

The relations between groups and monoids can be expressed with global theorem links. In Figure 2 they are indicated by dashed arrows.

The effect of global theorem links is the extension of the set of proved theorems available at a locale node. This is captured by the next definition.

Definition 7 *For a set T of global theorem links that are implied by the development graph $G = (N, L)$, the* extended set of proved theorems *is*

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

That is, proved theorems are accumulated via any conceivable path of definition and theorem links. The graph $(N, L \cup T)$ may contain cycles, and the set of paths to n is not necessarily finite. On the other hand, default sets of theorems for proof tools are required to be finite, hence so is the expansion $\mathrm{F}^*_{G,T}(n)$. By imposing a suitable restriction on the consequence morphisms, the extended set of proved theorems can be guaranteed to be finite.

Lemma 3 *Let $G = (N, L)$ be a development graph with the consequence relation $\vdash$ given by Isabelle's meta-logic. Let the consequence morphisms be substitutions that replace term variables by term variables. Their effect on type variables is unconstrained. Let T be a set of global theorem links implied by G. Then $\mathrm{F}^*_{G,T}(n)$ is finite for all $n \in G$.*

Proof. The set $\mathrm{F}^*_{G,T}(n)$ is the union of sets $\sigma(\mathrm{F}_G(k))$, and it has to be shown that their number is finite. Let $k \xrightarrow{\sigma}_* n \in L \cup T$. The morphism σ maps parameters of k to parameters of n. Since the number of global parameters of n is finite, so is the number of maps of global parameters of k to the global parameters of n. Each map defines a class of substitutions that have the same effect on the type and term parameters of k, and hence on $\mathrm{F}_G(k)$. This implies that there are only finitely many $\sigma(\mathrm{F}_G(k))$. The proof is complete since N is finite. □

The command **interpretation** $m \subseteq e$ asserts several global theorem links, which have to be proved. For each occurrence of a locale name n in the locale expression e, a link $n \xrightarrow{\sigma}_{\mathrm{T}} m$ is asserted. The expression defines a renaming, which in turn defines σ. Remember that the substitution on type parameters can be inferred. Each global theorem link gives rise to proof obligations. Which of these are implied by existing links?

Lemma 1 enables to decompose the global theorem links into an equivalent set of local theorem links. Let $G = (N, L)$ be the development graph and T be a set of global theorem links implied by G. A link $n \xrightarrow{\sigma}_{\mathrm{t}} m$ is implied by G if $n \xrightarrow{\sigma}_* m \in L \cup T$. On the other hand, if this is not the case, then the known links contain no information that would justify the new link. As consequence of the considerations in the proof of Lemma 3, only a finite number of paths have to be considered in order to test reachability. This implies that there is an algorithm to compute the proof obligations that cannot be deduced from the links.

4.2 Examples

As first set of examples on interpretation, we assert the global theorem links between the locales from Figure 2. Since existing theorem links are taken into account when computing proof obligations, the order in which these links are asserted influences which proof obligations are generated for each of the links. We start with

> **interpretation** lgroup $\subseteq$ rgroup.

This leads to the theorem link rgroup $\rightarrow_{\mathrm{T}}$ lgroup. By Lemma 1 it is decomposed into semigroup $\rightarrow_{\mathrm{t}}$ lgroup and rgroup $\rightarrow_{\mathrm{t}}$ lgroup. The first of these links is implied by the definition link of lgroup, thus it generates no proof obligation. The axioms of rgroup, $x \cdot_G 1_G = x$ and $x \cdot_G \mathrm{inv}_G\, x = 1_G$, remain to be shown, in the context of lgroup. Local theorem links and resulting proof obligations for the remaining three interpretations are shown in Table 1. Note that discharging the proof obligations for lgroup $\subseteq$ monoid is trivial, since both lone and rone are, at that stage, in the

Interpretation	Local theorem links	Proof obligations	
lgroup $\subseteq$ rgroup	semigroup $\to_{\mathrm{t}}$ lgroup	—	
	rgroup $\to_{\mathrm{t}}$ lgroup	$x \cdot_G 1_G = x$	$x \cdot_G \mathrm{inv}_G\, x = 1_G$
rgroup $\subseteq$ lgroup	semigroup $\to_{\mathrm{t}}$ rgroup	—	
	lgroup $\to_{\mathrm{t}}$ rgroup	$1_G \cdot_G x = x$	$\mathrm{inv}_G\, x \cdot_G x = 1_G$
lgroup $\subseteq$ monoid	semigroup $\to_{\mathrm{t}}$ lgroup	—	
	monoid $\to_{\mathrm{t}}$ lgroup	$x \cdot_G 1_G = x$	$1_G \cdot_G x = x$
rgroup $\subseteq$ monoid	semigroup $\to_{\mathrm{t}}$ rgroup	—	
	monoid $\to_{\mathrm{t}}$ rgroup	—	

Table 1: Asserting a sequence of interpretations between group locales.

extended set of proved theorems of lgroup. The interpretation merely turns this into a theorem link. The last interpretation, rgroup $\subseteq$ monoid, is unnecessary and generates no more proof obligations, since it is implied by the second and the third.

The next example probably serves no practical purpose in terms of its specification. Nevertheless it illustrates the nature of cycles in a development graph augmented by theorem links.

locale parity =
  fixes a **and** b **and** c
  assumes $a \leftrightarrow b \leftrightarrow c$

Its parameters are of type bool, and the specification says that an odd number of them must be true. The set of proved theorems $\mathrm{F}^*_G(\text{parity})$ is $\{a \leftrightarrow b \leftrightarrow c\}$. The locale admits

interpretation parity $\subseteq$ parity $b\ c\ a$,

which adds a loop to the set of global theorem links, where the morphism is a cyclic permutation of parameters. The extended set of proved theorems, $\mathrm{F}^*_{G,T}(\text{parity})$ is $\{a \leftrightarrow b \leftrightarrow c, b \leftrightarrow c \leftrightarrow a, c \leftrightarrow a \leftrightarrow b\}$, and it could be extended to all six permutations of the parameters by one additional interpretation.

The last example is about rings and polynomials. Assume that ring structures are represented with a record type and that the structure of polynomials is a subtype.[5] Then a locale that specifies rings has a single parameter. A functor UP, which is simply a function of the meta-logic, maps each ring structure R to the structure of univariate polynomials R[X]. If R is a ring then so is R[X]. We make the following declarations. Import graph and theorem links are shown in Figure 3.

locale up =
  fixes R **and** P
  defines $P \equiv$ UP R

[5] This restricts the example to Isabelle/HOL, because records are implemented only there.

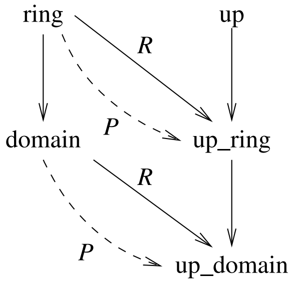

Figure 3: Dependencies between ring and polynomial ring locales.

> **locale** up_ring = up + ring R
> **interpretation** up_ring $\subseteq$ ring P $\quad\varphi$

The keyword **defines** is used to declare an axiom which has the form of a definition. In up the parameter P is defined to be a polynomial structure over R. In up_ring, R is specified to be a ring. With interpretation the ring theorems can be reused for polynomials. It requires the proof that polynomials over a ring form a ring, a rather long proof abbreviated by φ. Functors are monotonous on specifications. If the specification of R is strengthened to, say, integral domains, then, by incident, P becomes an integral domain as well. This can be adequately expressed by

> **locale** up_domain = up_ring + domain R
> **interpretation** up_domain $\subseteq$ domain P $\quad\varphi$

and the dependency management ensures that only the local axioms for domain P need to be proved explicitly, while the theorems of ring P are available in the context of this proof.

5 Conclusions

An important characteristic of theories in mathematics is that they are open. In the process of mathematical discovery, new theorems are added to them continuously. The same should be possible for the library of a prover, and it is required that the module system facilitates this; especially in the presence of complex module operations like interpretation. Development graphs [7] maintain module interdependencies explicitly and enable the automatic propagation of theorems — that is, without reprocessing (and reproving!) part of the theory hierarchy.

The modules of Coq [5] have a fixed signature and do not permit to add theorems dynamically. The situation is less clear for PVS [12]. Although there are no commands for the dynamic addition of theorems to theories, it might be possible to reprocess the theory hierarchy without reproving theorems of intermediate theories, since the system does not follow the LCF-approach. The system does not support cyclic interpretations.

Locales are based on the observation that Isabelle's meta-level sequents are elements of a consequence relation, and that (well-typed) substitutions are consequence morphisms. Hence import and interpretation dependencies can be managed with development graphs. These enable to discharge proof obligations that are implied by existing import and interpretations links. Hutter presents no completeness results. For locales, all proof obligations implied by the development graph can be discharged automatically.

The interpretation command presented here was implemented by the author and is available with Isabelle 2005. It has been used in the library of univariate polynomials (part of the Isabelle distribution as session HOL-Algebra). Previously 43 theorems on commutative rings, four theorems on integral domains and four theorems on algebras had to be lifted to polynomials manually, and likewise, in the context of the universal property, seven theorems on ring homomorphisms to the evaluation morphism. Thanks to interpretation, this is now automatic.

Development graphs were first implemented in the change management system Maya [2]. This is a high-level reasoner on top of an actual prover. It can maintain consistency between interdependent theories over changes made in a large-scale verification project. These changes include addition and deletion of axioms and import relations. Each change may invalidate theorems in theories and interpretation relations between theories. Maya determines which theorems and proof obligations have to be reproved. The change management we have implemented for locales is much simpler. It is only used to minimise the effort when establishing interpretations between locales.

The novelty of our work is that sets of proved theorems are computed automatically, and are made available in the context of proofs. This is of particular importance to support proof procedures. The sets of proved theorems are not necessarily finite, but they are for the class of consequence morphisms permitted by locales. Morphisms in IMPS [6] only enable to transport individual theorems between theories. Heuristics can help identify individual theorems together with suitable morphisms to aid particular proof situations.

The developers of Maya call the reasoning performed by the change management system *verification in the large* and the reasoning at the level of theorems *verification in the small* (see [2]). Maya connects both: reasoning in the large generates proof obligations to be discharged by reasoning in the small. Our integration takes this connection one step further: reasoning in the large is used to provide suitable contexts for reasoning in the small. This is particularly important in interactive proof. It is probably less important if the prover is mostly automatic, as is Inka [1], the prover connected to Maya.

We have argued that not only each proved theorem needs to be propagated, but also the control information attached to it — like the attribute simp, which adds a theorem to the set of rewrite rules used by the method simp. When locales are combined, also their sets of rewrite rules are combined, and this may have undesired effects. For example, the method simp might no longer terminate. This phenomenon is related to the combination of decision procedures, which is intrinsicly difficult. Isabelle's approach to this problem is pragmatic. An attribute for the deletion of theorems from the set of rewrite rules, for example, is provided. By demanding that the local set of proved theorems $\mathrm{F}_G(n)$ for a locale n is always considered last in the construction of the set of extended proved theorems $\mathrm{F}^*_{G,T}(n)$ the user is always in the position to make the required adjustments.

Acknowledgements. Dieter Hutter and Michael Kohlhase have pointed out the connection of dependency management of locales to development graphs. Using them has improved the presentation. Amine Chaieb, Tobias Nipkow, Tjark Weber and Makarius Wenzel have made useful comments on a draft of this paper.

References

[1] S. Autexier, D. Hutter, H. Mantel, and A. Schairer. System description: inka 5.0 — a logic voyager. In H. Ganzinger, editor, *Automated deduction — CADE-16, Trento, Italy*, LNCS 1632, pages 208–211. Springer, 1999.

[2] 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.

[3] C. Ballarin. Locales and locale expressions in Isabelle/Isar. In Berardi et al. [4], pages 34–50.

[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. 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.

[11] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. LNCS 2283. Springer, 2002.

[12] S. Owre and N. Shankar. Theory interpretation in PVS. Technical Report CSL-01-01, SRI, Apr. 2001.

[13] L. C. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5(3):363–397, 1989.

[14] B. L. van der Waerden. *Algebra I*. Springer-Verlag, 1960.