Chapter 5

Higher-order term indexing

Efficient term indexing techniques have resulted in dramatic speed improvements in all major first-order logic programming and theorem proving systems and have been crucial to their success. Broadly speaking, term indexing techniques facilitate the rapid retrieval of a set of candidate terms satisfying some property (e.g. unifiability, instance, variant etc.) from a large collection of terms. In logic programming, for example, we need to select all clauses from the program which unify with the current goal. In tabled logic programming we memoize intermediate goals in a table and reuse their results later in order to eliminate redundant and infinite computation. Here we need to find all entries in the table such that the current goal is a variant or an instance of a table entry and re-use the associated answers. If there is no such table entry, we need to add the current goal to the table. Since rapid retrieval and efficient storage of large collection of terms plays a central role in logic programming and in proof search in general, a variety of indexing techniques have been proposed for first-order terms (see [61] for a survey). However, indexing techniques for higher-order terms are missing thereby severely hampering the performance of higher-order systems such as Twelf [54], λProlog [40] or Isabelle [46].

In this chapter, we present a higher-order term indexing technique based on substitution trees. Substitution tree indexing [25] is a highly successful first-order term indexing strategy which allows the sharing of common sub-expressions via substitutions. Given the following two terms:

$$\mathsf{pred}\ (\mathsf{h}\ (\mathsf{g}\ \mathsf{a}))\ (\mathsf{g}\ \mathsf{b})\ \mathsf{a} \text{ and } \mathsf{pred}\ (\mathsf{h}\ (\mathsf{g}\ \mathsf{b}))\ (\mathsf{g}\ \mathsf{b})\ \mathsf{a}$$

we can compute the most specific generalization (msg) of both terms, which is

$$\mathsf{pred}\ (\mathsf{h}\ (\mathsf{g}\ *))\ (\mathsf{g}\ \mathsf{b})\ \mathsf{a}$$

where $*$ is a placeholder. Then we obtain the first term by substituting a for the placeholder and the second term by substituting b for it.

In this chapter we present an indexing technique for terms of the dependently typed lambda-calculus. This poses several challenges: First of all, building a substitution tree relies on computing the most specific generalization of two terms. However in the higher-order setting, the most specific generalization of two terms does not exist in general. Second, retrieving all terms, which unify or match, needs to be efficient – but higher-order unification is undecidable in general. As discovered by Miller [35], there exists a decidable fragment, called higher-order patterns. For this fragment, unification and computing the most specific generalization is decidable even in rich type theories with dependent types and polymorphism as shown by Pfenning [49]. However, these algorithms may not be efficient in practice [58] and hence it is not obvious that they are suitable for higher-order term indexing techniques.

In this chapter, we present substitution tree indexing for higher-order terms based on linear higher-order patterns (see Chapter 3). Linear higher-order patterns refine the notion of higher-order patterns further and factor out any computationally expensive parts. As we have shown in Chapter 3, many terms encountered fall into this fragment and linear higher-order pattern unification performs well in practice. In this chapter, we give algorithms for computing the most specific generalization of two linear higher-order patterns, for inserting terms in the index and for retrieving a set of terms from the index such that the query is an instance of the term in the index. This indexing structure is implemented as part of the Twelf system [54] to speed-up the execution of the tabled logic programming interpreter [56]. Experimental results show substantial performance improvements, between 100% and over 800%.

The chapter is organized as follows: In Section 5.1, we present the general idea of higher-order substitution trees. In Section 5.2, we give algorithms for computing the most specific generalization of two terms and inserting terms into the index. Retrieval is discussed in Section 5.3. In Section 5.4, we present some experimental results comparing the tabled higher-order logic programming interpreter with and without indexing.

5.1 Higher-order substitution trees

The general indexing problem can be described as follows: Given a large set $\mathcal{S}$ of indexed terms and a single term U called the *query term*, we have to retrieve quickly each term $V \in \mathcal{S}$ such that a *retrieval condition* R holds between U and V. In general the retrieval condition can be unifiability, instance or variant checking, or finding generalizations. The term indexing problem consists of finding a data-structure, called the *index*, which allows one to perform efficiently the following operations: 1) *index maintenance*, i.e. changing the index when terms are inserted (or deleted) from the set of indexed terms. 2) *term retrieval* i.e. finding some (or all) terms $V \in \mathcal{S}$ s.t. V and the query term U are in relation to each other.

To illustrate the notation, we give a sample signature together with a set of terms we want to index. We define a type family exp for expressions. In addition, we give constants a, b, c and constructors f, g, h for building expressions of this small language.

```
exp: type.
a: exp.                    h: exp -> exp.
b: exp.                    g: exp -> exp.
c: exp.                    f: (exp -> exp) -> exp.
```

Finally, we define a predicate pred which takes in three arguments, where each argument is an expression

```
pred: exp -> exp -> exp -> type.
```

Next, we give four examples of the predicate pred:

$$
\begin{array}{rll}
 & \mathsf{pred}\ (\mathsf{h}\ (\mathsf{g}\ \mathsf{b}))\ (\mathsf{g}\ \mathsf{b})\ \mathsf{a} & (1)\\
 & \mathsf{pred}\ (\mathsf{h}\ (\mathsf{g}\ \mathsf{b}))\ (\mathsf{g}\ \mathsf{b})\ \mathsf{b} & (2)\\
\Pi^{\Box}v{::}\cdot\vdash\mathsf{exp}. & \mathsf{pred}\ (\mathsf{h}\ (\mathsf{h}\ \mathsf{b}))\ (\mathsf{g}\ \mathsf{b})\ (\mathsf{f}\ \lambda x.v[\cdot]) & (3)\\
\Pi^{\Box}u{::}x{:}\mathsf{exp}\vdash\mathsf{exp}. & \mathsf{pred}\ (\mathsf{h}\ (\mathsf{h}\ \mathsf{c}))\ (\mathsf{g}\ \mathsf{b})\ (\mathsf{f}\ \lambda x.u[\mathsf{id}]) & (4)
\end{array}
$$

The first two predicates are closed, while predicate (3) and (4) contain existential variables v and u respectively which are bound by $\Pi^{\Box}$. Using these examples, we will briefly highlight some of the subtle issues concerning the interplay of bound and existential variables. The third predicate (3) refers to the existential variable v. Note that v is associated with the empty substitution, although it occurs in the context of a bound variable x. This means that any instantiation we find for v is not allowed to depend on the bound variable x. In contrast, predicate (4) refers to an existential variable u, which is associated with the identity substitution. This means that any instantiation we find for u may depend on the bound variable x. We come back to this point in Section 5.1.1. All these terms share some structural information. For example, the third argument is same in all the terms. When storing these terms, we would like to store the subterm $(\mathsf{g}\ \mathsf{b})$ only once. Similarly, when looking up if a query U is already in the index, we only want to check once if the third argument of U is $(\mathsf{g}\ \mathsf{b})$. In other words, an indexing data-structure should allow us to share common structure and common operation.

In this chapter, we will focus on substitution tree indexing. We start by giving the general idea higher-order substitution tree. To build a higher-order substitution tree, we proceed in two steps: First, we standardize the terms and convert terms into linear higher-order patterns. Second, we represent terms as a sequence of substitutions, which are stored in a tree.

5.1.1 Standardization: Linear higher-order pattern

To get the maximum structure sharing across different indexed terms it is important to use variables in a consistent manner. In first-order term indexing we therefore standardize the term before inserting a term into an index. In addition, first-order term indexing strategies often employ linearization. When converting the term into standard form every occurrence of an existential variable is represented as a distinct standardized existential variable. Together with the linear term, we then also store variable definitions, which establish the equality between these two variables. One of the reasons for using linearization is efficiency. Nonlinear terms may have multiple occurrences of the same existential variable and therefore requiring to check whether different substitutions for the same variable are consistent. Since such consistency checks can be potentially expensive and may lead to performance degradation, most first-order indexing techniques rely on a post-processing step to carry out the consistency checks.

To design a higher-order indexing technique, we will extend this notion of linearization and standardization. We will require that terms are linear higher-order patterns. Higher-order patterns [35, 49] are terms where every existential variable must be applied to *some* distinct bound variables. Linear higher-order patterns (see Chapter 3) impose some further restrictions on the structure of terms: First, all existential variables must occur only once. This allows us to delay any expensive consistency checks. Second, all existential variables must be applied to *all* distinct bound variables. This eliminates any computationally expensive checks involving bound variables. This observation to restrict higher-order patterns even further to patterns where existential variables must be applied to all bound variables has also been made by Hanus and Prehofer [26] in the context of higher-order functional logic programming. While Hanus and Prehofer syntactically disallow terms which are not fully applied, we translate any term into a linear higher-order pattern together with some variable definitions. Any consistency checks are delayed after linear higher-order pattern unification has succeeded. As we have shown in Chapter 3, performance of unification is improved substantially by this technique.

To illustrate, let us consider the previous term $\mathsf{pred}\ (\mathsf{h}\ (\mathsf{g}\ \mathsf{b}))\ (\mathsf{g}\ \mathsf{b})\ (\mathsf{f}\ \lambda x.v[\cdot])$. The existential variable $v[\cdot]$ occurs only once, but is not applied to all distinct bound variables, since v is not allowed to depend on the bound variable x. Therefore $v[\cdot]$ is a higher-order pattern, but it is not linear. We can enforce that every existential variable occurs only once and is applied to *all* bound variables, by translating it into a linear higher-order pattern:

$$\mathsf{pred}\ (\mathsf{h}\ (\mathsf{g}\ \mathsf{b}))\ (\mathsf{g}\ \mathsf{b})\ (\mathsf{f}\ \lambda x.u[x/x])$$

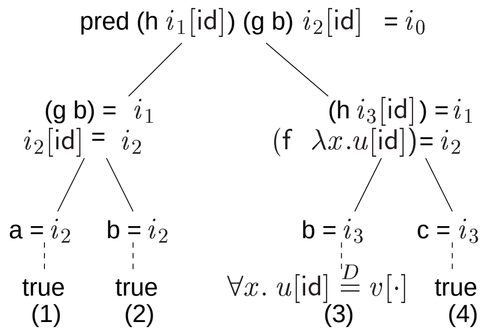
together with a variable definition,

$$\forall x{:}\mathsf{exp}.u[x/x] \overset{D}{=} v[\cdot]$$

where u is a new existential variable which is applied to the bound variable x. Similarly to first-order standardization and linearization, linearization in the higher-order case remains quite straightforward and can be done by traversing the term once. The main difference is that the postponed variable definitions may be more complex, as we have discussed earlier in Chapter 3.

5.1.2 Example

In this section, we show the substitution tree for the set of terms given earlier. In each node, we store a set of substitutions, which we write here as $U = i$ where i is an internal modal variable and i will always be applied to all bound variables it may depend on. Therefore it will always be associated with the identity substitution id.



Note that variable definitions are stored at the leafs. By composing the substitutions in the right-most branch for example we get

$$[\![\mathsf{c}/i_3]\!][\![\mathsf{f}\ \lambda x.\ u[\mathsf{id}]/i_2,\ (\mathsf{h}\ i_3[\mathsf{id}])/i_1]\!][\![\mathsf{pred}\ (\mathsf{h}\ i_1[\mathsf{id}])\ (\mathsf{g}\ \mathsf{b})\ i_2[\mathsf{id}]/i_0]\!]i_0[\mathsf{id}]$$

which represents the term (4) $\mathsf{pred}\ (\mathsf{h}\ (\mathsf{h}\ \mathsf{c}))\ (\mathsf{g}\ \mathsf{b})\ (\mathsf{f}\ \lambda x.\ u[\mathsf{id}])$. Note that in the implementation we can omit the identity substitution $i_2[\mathsf{id}]/i_2$ and maintain as an invariant that the substitutions in each node can be extended appropriately. This will simplify the theoretical treatment slightly.

In the definition of higher-order substitution trees we will distinguish between a modal context Δ which denotes the original modal variables and a modal context Σ for the internal modal variables. Note that the modal variables in Σ have no dependencies among each other and can be arbitrarily re-ordered and may only depend on variables in Δ. Moreover, every internal modal variable i will be created in such a way that it is applied to all bound variables, i.e. it will be associated with the identity substitution id.

Definition 66 (Higher-order substitution tree) *A higher-order substitution tree is an ordered tree.*

1. *A tree is a leaf node with substitution ρ such that $\Delta \vdash (\mathsf{id}_\Delta, \rho) : (\Delta, \Sigma)$.*

2. *A tree is a node with substitution ρ such that $(\Delta, \Sigma) \vdash \rho : (\Delta, \Sigma')$ and children nodes $N_1, \ldots, N_n$ where each child Node N_i has a substitution ρ_i such that $(\Delta, \Sigma_i) \vdash (\mathsf{id}_\Delta, \rho_i) : (\Delta, \Sigma)$.*

Moreover, for every path from the top node ρ_0 where $(\Delta, \Sigma_1) \vdash \rho_0 : (\Delta, \Sigma_0)$ to the leaf node ρ_n, we have $\Delta \vdash [\![\rho_n]\!]([\![\rho_{n-1}]\!] \ldots \rho_0) : (\Delta, \Sigma_0)$. In other words, there are no internal modal variables left after we compose all the substitutions ρ_n up to ρ_0.

Inserting a new term U into the substitution tree corresponds to inserting the substitution $U = i_0$. Before presenting the algorithms for building substitution trees, we discuss our reasons for adapting substitution tree indexing to the higher-order setting rather than for example discrimination tries. First, we note that the order of term traversal is not fixed in advance. For example, in the substitution tree given earlier we compare the substitution for the third argument before the substitutions for the first argument when looking up the term (3) and term (4). While we traverse the term (1) and (2) from left to right. This feature leads to very compact substitution trees and better memory usage and retrieval times. However, in general there may be multiple ways to insert a term and no optimal substitution trees exist. In contrast to other indexing techniques such as discrimination tries, substitution trees allows the sharing of common sub-expressions rather than common prefixes.

This is especially important for indexing dependently typed terms. To illustrate this point, we define a data-structure for lists consisting of characters and we keep track of the size of the list by using dependent types.

char : type.	list : char $\rightarrow$ type.
a : char.	nil : list 0.
b : char.	cons : $\Pi^{\square} n$::int .char $\rightarrow$ list $n \rightarrow$ list $(n+1)$.
test : $\Pi^{\square} n$::int .list $n \rightarrow$ type.	

The size of lists is an explicit argument to the predicate test. Hence test takes in two arguments, the first one is the size of the list and the second one is the actual list. The list constructor cons takes in three arguments. The first one denotes the size of the list, the second argument denotes the head and the third one denotes the tail. To illustrate, we give a few examples. We use gray color for the explicit arguments.

test 4 (cons 4 a (cons 3 a (cons 2 a (cons 1 b nil))))
test 5 (cons 5 a (cons 4 a (cons 3 a (cons 2 a (cons 1 b nil)))))
test 6 (cons 6 a (cons 5 a (cons 4 a (cons 3 b (cons 2 a (cons 1 b nil))))))

If we use non-adaptive indexing techniques such as discrimination tries, we process the term from left to right and we will be able to share common prefixes. In the given example, such a technique discriminates on the first argument, which denotes the size of the list and leads to no sharing between the second argument. The substitution tree on the other hand allows us to share the structure of the second argument. The most specific generalization in this example is

test i_1[id] (cons i_2[id] a (cons i_3[id] a (cons i_4[id] a (cons i_5[id] i_6[id] nil)))).

This allows us to skip over the implicit first argument denoting the size and indexing on the second argument, the actual list. It has been sometimes argued that it is possible to retain the flexibility in non-adaptive indexing techniques by reordering the arguments to test. However, this only works easily in an untyped setting and it is not clear how to maintain typing invariants in a dependently typed setting if we allow arbitrary reordering of arguments. Hence higher-order substitution trees offer a adaptive compact indexing data-structure while maintaining typing invariants.

5.2 Insertion

Insertion of a term U into the index is viewed as insertion of the substitution U/i_0. Assuming that U has type A in a modal context Δ and a bound variable context Γ, U/i_0 is a modal substitution such that $\Delta \vdash U/i_0 : i{::}(\Gamma{\vdash}A)$. We will call the modal substitution which is going to be inserted into the index ρ. It is often convenient to consider the extended modal substitution $\Delta \vdash \mathsf{id}_\Delta, U/i_0 : (\Delta, i{::}(\Gamma{\vdash}A))$. This will simplify the following theoretical development. Again we note that in an implementation, we do not need to carry around explicitely the modal substitution id_Δ, but can always assume that any substitution can be extended appropriately.

The insertion process works by following down a path in the tree that is *compatible* with the modal substitution ρ. To formally define insertion, we show how to compute the most specific linear generalization (msg) of two modal substitutions and describe the most specific linear generalization of two terms. The algorithm for insertion will be defined for types, where all the dependencies have been erased (see Chapter 2). In Chapter 2, we have used erasure of type dependencies to define an algorithm for structurally equality of normal objects. Moreover, we have shown that if two normal objects are structurally equal then they are definitionally equal. Here we apply this central idea of erasing type dependencies to define an algorithm for computing the most specific generalization and show correctness of the algorithm. Instead of the strong typing invariants used in the previous definition for linear substitution trees, we will only require that the substitutions are "approximately" well-typed where all the dependencies have been erased.

We start by giving the judgements for computing the most specific generalization of two modal substitutions.

$$(\Delta, \Sigma_1)^- \vdash \rho_1 \sqcup \rho_2 : \Sigma_2^- \Longrightarrow \rho/(\Sigma^-, \theta_1, \theta_2) \quad \rho \text{ is the msg of } \rho_1 \text{ and } \rho_2$$

If $(\Delta, \Sigma_1) \vdash (\mathsf{id}_\Delta, \rho_1) : (\Delta, \Sigma_2)$ and $(\Delta, \Sigma_1) \vdash (\mathsf{id}_\Delta, \rho_2) : (\Delta, \Sigma_2)$ then ρ is the msg of ρ_1 and ρ_2. Moreover, ρ is a modal substitution such that $[\![\mathsf{id}_\Delta, \theta_1]\!]\rho$ is structurally equal to ρ_1 and $[\![\mathsf{id}_\Delta, \theta_2]\!]\rho$ is structurally equal to ρ_2. We think of ρ_1 as the modal substitution which is already in the index, while the modal substitution ρ_2 is to be inserted. As a consequence, only ρ_1 will refer to the internal modal variables in Σ_1, while ρ_2 only depends on the modal variables in Δ. The result of the msg are the modal substitution θ_1 and θ_2, where $\Delta, \Sigma_1 \vdash \theta_1 : \Sigma$ and $\Delta, \Sigma_1 \vdash \theta_2 : \Sigma$. In other words, θ_1 (resp. θ_2) only replaces internal modal variables in Σ. Note that any modal substitution ρ or θ with domain Σ, can be extended to a modal substitution $(\mathsf{id}_\Delta, \rho)$ (or $(\mathsf{id}_\Delta, \theta)$ resp.) with domain (Δ, Σ).

First, we give the rules for computing the most specific generalization of two modal substitutions. We use the abbreviations introduced in Chapter 2 and refer to the modal context Δ^- and Σ^- where all the type dependencies have been erased as Λ and Θ. The bound variable context Γ^- is abbreviated with Ω.

$$\frac{}{(\Lambda, \Theta) \vdash \cdot : \cdot \Longrightarrow \cdot/(\cdot, \cdot, \cdot)}$$

$$\frac{(\Lambda, \Theta_1) \vdash \rho_1 \sqcup \rho_2 : \Theta_2 \Longrightarrow \rho/(\Theta, \theta_1, \theta_2) \quad (\Lambda, \Theta_1); \Omega \vdash L_1 \sqcup L_2 : \tau \Longrightarrow L/(\Theta', \theta_1', \theta_2')}{(\Lambda, \Theta_1) \vdash (\rho_1, L_1/i) \sqcup (\rho_2, L_2/i) : (\Theta_2, i{::}(\Omega{\vdash}\tau)) \Longrightarrow (\rho, L/i)/((\Theta, \Theta'), (\theta_1, \theta_1'), (\theta_2, \theta_2'))}$$

Note, that we are allowed to just combine the modal substitutions θ_1 (θ_2 resp.) and θ_1' (θ_2' resp.) since we require that they refer to distinct modal variables and all the modal variables occur uniquely. Computing the most specific generalization of two modal substitutions relies on finding the most specific generalization of two objects. Note that we require that all objects are linear higher-order patterns and are in quasi-canonical form (see Chapter 2). Quasi-canonical forms are $\beta\eta$-normal forms where the type information on the bound variables in lambda-abstraction may be omitted. Moreover, we assume that all modal variables are lowered and have atomic type. We will use the spine notation introduced earlier in Chapter 4.

$$(\Delta, \Sigma)^-; \Gamma^- \vdash L_1 \sqcup L_2 : A^- \quad \Longrightarrow \quad L/(\Sigma'^-,\ \theta_1, \theta_2) \quad L \text{ is the msg of } L_1 \text{ and } L_2$$

$$(\Delta, \Sigma)^-; \Gamma^- \vdash S_1 \underline{\sqcup} S_2 : A^- > a^- \quad \Longrightarrow \quad S/(\Sigma'^-,\ \theta_1, \theta_2) \quad S \text{ is the msg of } S_1 \text{ and } S_2$$

If the terms L_1 and L_2 have type A in modal context (Δ, Σ) and bound variable context Γ, then L is the most specific generalization of L_1 and L_2 such that $[\![\mathsf{id}_\Delta, \theta_1]\!]L$ is structurally equal to L_1 and $[\![\mathsf{id}_\Delta, \theta_2]\!]L$ is structurally equal to L_2. Moreover, θ_1 and θ_2 are modal substitutions which map modal variables from Σ'^- to the modal context $(\Delta, \Sigma)^-$. For spines a similiar invariant holds. If S_1 and S_2 are spines from heads of type A to terms of type a, then S is the msg of S_1 and S_2 such that $[\![\mathsf{id}_\Delta, \theta_1]\!]S$ is structurally equal to S_1 and $[\![\mathsf{id}_\Delta, \theta_2]\!]S$ is structurally equal to S_2. θ_1 and θ_2 are modal substitutions which map modal variables from Σ'^- to the modal context $(\Delta, \Sigma)^-$.

We think of L_1 (or S_1)as an object which is already in the index and L_2 (or S_2) is the object to be inserted. As a consequence, only L_1 (and S_1) may refer to the internal variables in Σ, while L_2 (and S_2) only depends on Δ. The inference rules for computing the msg are given next.

Normal linear objects

$$\frac{(\Lambda, \Theta); \Omega, x{:}\tau_1 \vdash L_1 \sqcup L_2 : \tau_2 \Longrightarrow L/(\Theta', \theta_1, \theta_2)}{(\Lambda, \Theta); \Omega \vdash \lambda x.L_1 \sqcup \lambda x.L_2 : \tau_1 \to \tau_2 \Longrightarrow \lambda x.L/(\Theta',\ \theta_1, \theta_2)}$$

$$\frac{u{::}(\Phi \vdash \tau) \in \Lambda}{(\Lambda, \Theta); \Omega \vdash u[\pi] \sqcup u[\pi] : \tau \Longrightarrow u[\pi]/(\cdot, \cdot, \cdot)}\ (*)$$

$$\frac{u{::}\Phi \vdash \tau \in (\Lambda, \Theta) \quad i \text{ must be new}}{(\Lambda, \Theta); \Omega \vdash u[\pi] \sqcup L : \tau \Longrightarrow i[\mathsf{id}_\Omega]/(i{::}\Omega \vdash \tau,\ u[\pi]/i,\ L/i)}\ (1a)$$

$$\frac{i{::}\Omega \vdash \tau \in (\Lambda, \Theta)}{(\Lambda, \Theta); \Omega \vdash i[\mathsf{id}_\Omega] \sqcup L : \tau \Longrightarrow i[\mathsf{id}_\Omega]/(i{::}\Omega \vdash \tau,\ i[\mathsf{id}_\Omega]/i,\ L/i)}\ (1b)$$

$$\frac{u{::}(\Phi \vdash \tau) \in \Lambda}{(\Lambda, \Theta); \Omega \vdash L \sqcup u[\pi] : \tau \Longrightarrow i[\mathsf{id}_\Omega]/(i{::}\Omega \vdash \tau,\ L/i,\ u[\pi]/i)}\ (2)$$

$$\frac{(\Lambda, \Theta); \Omega \vdash S_1 \sqcup\!\!\!\!\_\ S_2 : \tau > \alpha \Longrightarrow S/(\Theta', \theta_1, \theta_2)}{(\Lambda, \Theta); \Omega \vdash H \cdot S_1 \sqcup H \cdot S_2 : \alpha \Longrightarrow H \cdot S/(\Theta', \theta_1, \theta_2)}\ (3a)$$

$$\frac{H_1 \neq H_2 \quad i \text{ must be new}}{(\Lambda, \Theta); \Omega \vdash H_1 \cdot S_1 \sqcup H_2 \cdot S_2 : \alpha \Longrightarrow i[\mathsf{id}_\Omega]/((i{::}\Omega \vdash \alpha), (H_1 \cdot S_1/i), (H_2 \cdot S_2/i))}\ (3b)$$

Normal linear spines

$$\frac{}{(\Lambda, \Theta); \Omega \vdash \mathsf{nil} \sqcup\!\!\!\!\_\ \mathsf{nil} : \alpha > \alpha \Longrightarrow \mathsf{nil}/(\cdot, \cdot, \cdot)}$$

$$\frac{\begin{array}{c}(\Lambda, \Theta); \Omega \vdash L_1 \sqcup L_2 : \tau_1 \Longrightarrow U/(\Theta_1,\ \theta_1, \theta_1') \\ (\Lambda, \Theta); \Omega \vdash S_1 \sqcup\!\!\!\!\_\ S_2 : \tau_2 > \alpha \Longrightarrow S/(\Theta_2,\ \theta_1, \theta_2')\end{array}}{(\Lambda, \Theta); \Omega \vdash (L_1; S_1) \sqcup\!\!\!\!\_\ (L_2; S_2) : \tau_1 \to \tau_2 > \alpha \Longrightarrow (L; S)/((\Theta_1, \Theta_2), (\theta_1, \theta_2), (\theta_2', \theta_1')}\ (4)$$

Note in the rule for lambda, we do not need to worry about capture, since modal variables and bound variables are defined in different context. Rule (*) treats the case where both terms are existential variables. Note that we require that both existential variables must be the same and their associated substitutions must also be equal. In rule (1) and (2), we just create the substitution $u[\pi]/i$. In general, we would need to create $[\mathsf{id}_\Omega]^{-1}(u[\pi])$, but since we know that π is a permutation substitution, we know that $[\mathsf{id}_\Omega]^{-1}(\pi)$ exists. In addition the inverse substitution of the identity is the identity. Note that we distinguish between the internal modal variables i and the "global" modal variables u in the rules (1a) and (1b). The key distinction is that we pick a new internal modal variable i in rule (1a) while we re-use the internal modal variable i in rule (1b). This is important for maintaining the invariant that the any child of $(\Lambda, \Theta_2) \vdash (\mathsf{id}_\Lambda, \rho) : (\Lambda, \Theta_1)$ has the form $(\Lambda, \Theta_3) \vdash (\mathsf{id}_\Lambda, \rho') : (\Lambda, \Theta_2)$ during insertion (see the insertion algorithm later on).

In rule (3a) and (3b) we distinguish on the head symbol H and compute the most specific generalization of two objects $H_1 \cdot S_1$ and $H_1 \cdot S_1$. If H_1 and H_2 are not equal, then we generate a new internal modal variable $i[\mathsf{id}_\Omega]$ together with the substitutions $H_1 \cdot S_1/i$ and $H_2 \cdot S_2/i$. Otherwise, we traverse the spines S_1 and S_2 and compute the most specific generalization of them. In rule (4), we can just combine the substitution θ_1 and θ_2, as we require that all modal variables occur uniquely, and hence there are no dependencies among Σ_1 and Σ_2.

Definition 67 (Compatibility of normal objects) *Two normal objects L_1 and L_2 are not compatible, if $(\Lambda, \Theta); \Omega \vdash L_1 \sqcup L_2 : \tau \Longrightarrow i[\mathsf{id}_\Omega]/(i{::}\Omega{\vdash}\tau, L_1/i, L_2/i)$. Otherwise, we call L_1 and L_2 compatible.*

In other words, we call two terms compatible, if they share at least the head symbol or a λ-prefix. Moreover, if two terms are incompatible then they must be syntactially different. Similarly, we can define the compatibility of two substitutions.

Definition 68 (Compatibility of modal substitutions)
Two modal substitutions $(\Delta, \Sigma_1) \vdash \rho_1 : \Sigma_2$ and $(\Delta, \Sigma) \vdash \rho_2 : \Sigma_2$ are not compatible, if $(\Delta, \Sigma_1)^- \vdash \rho_1 \sqcup \rho_2 : \Sigma_2^- \Longrightarrow \mathsf{id}_\Theta/(\Theta, \rho_1, \rho_2)$. Otherwise, we call ρ_1 and ρ_2 compatible.

As a consequence, if ρ_1 and ρ_2 are incompatible, then for any $L_1/i \in \rho_1$ and $L_2/i \in \rho_2$, we know that L_1 and L_2 are incompatible.

Next, we prove some lemmas which allow us to extend modal substitutions.

Lemma 69

1. *Let Σ be the context for all the modal variables occuring in σ and all elements in Σ are independent of each other.*
   If $\Delta \vdash \theta : \Sigma$, and $\Delta \vdash (\theta, \theta') : (\Sigma, \Sigma')$ then $[\![\theta]\!]\sigma = [\![\theta, \theta']\!]\sigma$ and

2. *Let Σ be the context for all the modal variables occuring in L and all elements in Σ are independent of each other.*
   If $\Delta \vdash \theta : \Sigma$, and $\Delta \vdash (\theta, \theta') : (\Sigma, \Sigma')$ then $[\![\theta]\!]L = [\![\theta, \theta']\!]L$.

3. *Let Σ be the context for all the modal variables occuring in S and all elements in Σ are independent of each other.*
   If $\Delta \vdash \theta : \Sigma$, and $\Delta \vdash (\theta, \theta') : (\Sigma, \Sigma')$ then $[\![\theta]\!]S = [\![\theta, \theta']\!]S$.

Proof: Induction on the structure of σ, L and S. □

Lemma 70

1. *Let Σ be the context for all the modal variables occuring in σ and all elements in Σ are independent of each other.*
   If $\Delta \vdash \theta : \Sigma$, and $\Delta \vdash (\theta', \theta) : (\Sigma', \Sigma)$ then $[\![\theta]\!]\sigma = [\![\theta', \theta]\!]\sigma$ and

2. *Let Σ be the context for all the modal variables occuring in L and all elements in Σ are independent of each other.*
   If $\Delta \vdash \theta : \Sigma$, and $\Delta \vdash (\theta', \theta) : (\Sigma', \Sigma)$ then $[\![\theta]\!]L = [\![\theta', \theta]\!]L$.

3. *Let Σ be the context for all the modal variables occuring in S and all elements in Σ are independent of each other.*
   If $\Delta \vdash \theta : \Sigma$, and $\Delta \vdash (\theta', \theta) : (\Sigma', \Sigma)$ then $[\![\theta]\!]S = [\![\theta, \theta']\!]S$.

Proof: Induction on the structure of σ, L and R. □

[1]

[1] NOTE: this isn't quite correct – when we apply the lemma, we only know that the results are approximately well-typed. ...

Theorem 71 (Soundness)

1. *If* $(\Delta,\Sigma)^-;\Gamma^- \vdash L_1 \sqcup L_2 : A^- \Longrightarrow L/(\Sigma'^-,\theta_1,\theta_2)$ *and*
   $(\Delta,\Sigma);\Gamma \vdash L_1 : A$ *and* $(\Delta,\Sigma);\Gamma \vdash L_2 : A$
   then $(\Delta,\Sigma) \vdash \theta_1 : \Sigma'$ *and* $(\Delta,\Sigma) \vdash \theta_2 : \Sigma'$ *and*
   $(\Delta,\Sigma)^-;\Gamma^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_1]\!]L : A^-$ *and* $(\Delta,\Sigma)^-;\Gamma^- \vdash L_2 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_2]\!]L : A^-$.

2. *If* $(\Delta,\Sigma)^-;\Gamma^- \vdash S_1 \underline{\sqcup} S_2 : A^- \Longrightarrow S/(\Sigma'^-,\theta_1,\theta_2)$ *and*
   $(\Delta,\Sigma);\Gamma \vdash S_1 : A$ *and* $(\Delta,\Sigma);\Gamma \vdash S_2 : A$
   then $(\Delta,\Sigma) \vdash \theta_1 : \Sigma'$ *and* $(\Delta,\Sigma) \vdash \theta_2 : \Sigma'$ *and*
   $(\Delta,\Sigma)^-;\Gamma^- \vdash S_1 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_1]\!]S : A^-$ *and* $(\Delta,\Sigma)^-;\Gamma^- \vdash S_2 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_2]\!]S : A^-$.

Proof: Simultanous induction on the structure of the first derivation. We give here a few cases.

Case $\mathcal{D} = (\Delta,\Sigma)^-;\Gamma^- \vdash \lambda x.L_1 \sqcup \lambda x.L_2 : A^- \to B^- \Longrightarrow \lambda x.L/(\Sigma'^-,\theta_1,\theta_2)$

$A^- \to B^- = (\Pi x{:}A.B)^-$	by erasure property
$(\Delta,\Sigma)^-;\Gamma^-,x{:}A^- \vdash L_1 \sqcup L_2 : B^- \Longrightarrow L/(\Sigma'^-,\theta_1,\theta_2)$	by inversion
$(\Delta,\Sigma);\Gamma \vdash \lambda x.L_1 : \Pi x{:}A.B$	by assumption
$(\Delta,\Sigma);\Gamma,x{:}A \vdash L_1 : B$	by inversion lemma
$(\Delta,\Sigma);\Gamma \vdash \lambda x.L_2 : \Pi x{:}A.B$	by assumption
$(\Delta,\Sigma);\Gamma,x{:}A \vdash L_2 : B$	by inversion lemma
$(\Delta,\Sigma)^- \vdash \theta_1 : \Sigma'^-$	by ih
$(\Delta,\Sigma)^- \vdash \theta_2 : \Sigma'^-$	by ih
$(\Delta,\Sigma)^-;\Gamma^-,x{:}A^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_1]\!]L : B^-$	by ih
$(\Delta,\Sigma)^-;\Gamma^- \vdash \lambda x.L_1 \longleftrightarrow \lambda x.[\![\mathsf{id}_\Delta,\theta_1]\!]L : A^- \to B^-$	by rule
$(\Delta,\Sigma)^-;\Gamma^- \vdash \lambda x.L_1 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_1]\!](\lambda x.L) : A^- \to B^-$	by modal subst. definition
$(\Delta,\Sigma)^-;\Gamma^-,x{:}A^- \vdash L_2 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_2]\!]L : B^-$	by ih
$(\Delta,\Sigma)^-;\Gamma^- \vdash \lambda x.L_2 \longleftrightarrow \lambda x.[\![\mathsf{id}_\Delta,\theta_2]\!]L : A^- \to B^-$	by rule
$(\Delta,\Sigma)^-;\Gamma^- \vdash \lambda x.L_2 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_2]\!](\lambda x.L) : A^- \to B^-$	by modal subst. definition

Case $\mathcal{D} = (\Delta,\Sigma)^-;\Gamma^- \vdash u[\pi] \sqcup u[\pi] : A^- \Longrightarrow u[\pi]/(\cdot,\cdot,\cdot)$

$u{::}(\Psi{\vdash}A) \in \Delta$ by inversion
$(\Delta,\Sigma);\Gamma \vdash u[\pi] : A$ by assumption
$(\Delta,\Sigma)^-;\Gamma^- \vdash u[\pi] \longleftrightarrow u[\pi] : A^-$ by reflexivity
$(\Delta,\Sigma)^- \vdash \cdot : \cdot$ by rule

Case $\mathcal{D} = (\Delta,\Sigma)^-;\Gamma^- \vdash u[\pi] \sqcup L : A^- \Longrightarrow i[\mathsf{id}_{\Gamma^-}]/(i{::}\Gamma^-{\vdash}A^-,\ u[\pi]/i,\ L/i)$

$u{::}(\Psi{\vdash}A) \in \Delta$ by inversion
$(\Delta,\Sigma);\Gamma \vdash u[\pi] : A$ by assumption
$(\Delta,\Sigma);\Gamma \vdash L : A$ by assumption
$(\Delta,\Sigma)^-;\Gamma^- \vdash u[\pi] \longleftrightarrow [\![\mathsf{id}_\Delta, u[\pi]/i]\!]i[\mathsf{id}_{\Gamma^-}] : A^-$
$(\Delta,\Sigma)^-;\Gamma^- \vdash u[\pi] \longleftrightarrow u[\pi] : A^-$ by reflexivity
$(\Delta,\Sigma)^-;\Gamma^- \vdash L \longleftrightarrow [\![\mathsf{id}_\Delta, L/i]\!]i[\mathsf{id}_{\Gamma^-}] : A^-$
$(\Delta,\Sigma)^-;\Gamma^- \vdash L \longleftrightarrow L : A^-$ by reflexivity
$(\Delta,\Sigma) \vdash L/i : i{::}\Gamma{\vdash}A$ by rule using assumption
$(\Delta,\Sigma) \vdash u[\pi]/i : i{::}\Gamma{\vdash}A$ by rule using assumption

Case $\mathcal{D} = (\Delta,\Sigma)^-;\Gamma^- \vdash H \cdot S_1 \sqcup H \cdot S_2 : a^- \Longrightarrow H \cdot S/((\Sigma_1,\Sigma_2),\ (\theta_1,\theta_1'),\ (\theta_2,\theta_2'))$

$(\Delta,\Sigma)^-;\Gamma^- \vdash S_1 \underline{\sqcup} S_2 : A^- > a^- \Longrightarrow S/(\Sigma_1,\theta_1,\theta_2)$
$(\Delta,\Sigma);\Gamma \vdash H \cdot S_1 : a$ by assumption
$(\Delta,\Sigma);\Gamma \vdash S_1 : A > a$ by inversion
$(\Delta,\Sigma);\Gamma \vdash H \cdot S_2 : a$ by assumption
$(\Delta,\Sigma);\Gamma \vdash S_2 : A > a$ by inversion
$(\Delta,\Sigma)^-;\Gamma^- \vdash S_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_1]\!]S : A^- > a^-$ by ih
$(\Delta,\Sigma)^-;\Gamma^- \vdash S_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_2]\!]S : A^- > a$ by ih

$(\Delta, \Sigma) \vdash \theta_1 : \Sigma_1$ by ih
$(\Delta, \Sigma) \vdash \theta_2 : \Sigma_1$ by ih
$(\Delta, \Sigma) \vdash \theta'_1 : \Sigma_2$ by ih
$(\Delta, \Sigma) \vdash \theta'_2 : \Sigma_2$ by ih
$(\Delta, \Sigma) \vdash (\theta_1, \theta'_1) : (\Sigma_1, \Sigma_2)$ since θ_1 and θ'_1 refer to distinct modal variables
$(\Delta, \Sigma) \vdash (\theta_2, \theta'_2) : \Sigma_1, \Sigma_2)$ since θ_2 and θ'_2 refer to distinct modal variables
$(\Delta, \Sigma)^-; \Gamma^- \vdash S_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_1, \theta'_1]\!]S : A^- > a^-$ by lemma 69
$(\Delta, \Sigma)^-; \Gamma^- \vdash S_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_2, \theta'_2]\!]S : A^- > a^-$ by lemma 69
$(\Delta, \Sigma)^-; \Gamma^- \vdash H \cdot S_1 \longleftrightarrow H \cdot\ [\![\mathsf{id}_{\Delta^-}, \theta_1, \theta'_1]\!]S : a^-$ by rule
$(\Delta, \Sigma)^-; \Gamma^- \vdash H \cdot S_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_1, \theta'_1]\!](H \cdot S) : a^-$ by mod subst definition
$(\Delta, \Sigma)^-; \Gamma^- \vdash H \cdot S_2 \longleftrightarrow H \cdot\ [\![\mathsf{id}_{\Delta^-}, \theta_2, \theta'_2]\!]S : a^-$ by rule
$(\Delta, \Sigma)^-; \Gamma^- \vdash H \cdot S_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_2, \theta'_2]\!](H \cdot\ S) : a^-$ by mod subst definition

Case $\mathcal{D} = (\Delta, \Sigma)^-; \Gamma^- \vdash H_1{\cdot}S_1 \sqcup H_2{\cdot}S_2 : a^- \Longrightarrow i[\mathsf{id}_\Gamma]/(i{::}\Gamma^- \vdash a^-,\ H_1{\cdot}S_1/i,\ H_2{\cdot}S_2/i)$

$(\Delta, \Sigma); \Gamma \vdash H_1 \cdot S_1 : a$ by assumption
$(\Delta, \Sigma); \Gamma \vdash H_2 \cdot S_2 : a$ by assumption
$(\Delta, \Sigma)^-; \Gamma^- \vdash H_1 \cdot S_1 \longleftrightarrow [\![(H_1 \cdot S_1)/i]\!](i[\mathsf{id}_{\Gamma^-}]) : a^-$ by modal subst. def
$(\Delta, \Sigma)^-; \Gamma^- \vdash H_1 \cdot S_1 \longleftrightarrow H_1 \cdot S_1 : a^-$ by reflexivity
$(\Delta, \Sigma)^-; \Gamma^- \vdash H_2 \cdot S_2 \longleftrightarrow [\![H_2 \cdot S_2/i]\!](i[\mathsf{id}_{\Gamma^-}]) : a^-$ by modal subst. def
$(\Delta, \Sigma)^-; \Gamma^- \vdash H_2 \cdot S_2 \longleftrightarrow H_2 \cdot S_2 : a^-$ by reflexivity

Case $\mathcal{D} = (\Delta, \Sigma)^-; \Gamma^- \vdash (L_1; S_1) \mathrel{\underline{\sqcup}} (L_2; S_2) : (A^- \to B^-) > a^-$
$\Longrightarrow (L; S)/((\Sigma_1, \Sigma_2),\ (\theta_1, \theta'_1),\ (\theta_2, \theta'_2))$

$(\Pi x{:}A.B)^- = A^- \to B^-$ by erasure property
$(\Delta, \Sigma)^-; \Gamma^- \vdash L_1 \sqcup L_2 : A^- \Longrightarrow L/(\Sigma_1,\ \theta_1,,\ \theta_2)$ by inversion
$(\Delta, \Sigma)^-; \Gamma^- \vdash S_1 \mathrel{\underline{\sqcup}} S_2 : B^- > a^- \Longrightarrow S/(\Sigma_2,\ \theta'_1,,\ \theta'_2)$
$(\Delta, \Sigma); \Gamma \vdash (L_1; S_1) : \Pi x{:}A.B > a^-$ by assumption
$(\Delta, \Sigma); \Gamma \vdash L_1 : A$ by inversion
$(\Delta, \Sigma); \Gamma \vdash S_1 : [L_1/x]B > a$
$([L_1/x]B)^- = B^-$ by erasure property

$(\Delta,\Sigma);\Gamma \vdash (L_2;S_2) : \Pi x{:}A.B > a^-$ by assumption
$(\Delta,\Sigma);\Gamma \vdash L_2 : A$ by inversion
$(\Delta,\Sigma);\Gamma \vdash S_2 : [L_2/x]B > a$
$([L_2/x]B)^- = B^-$ by erasure property
$(\Delta,\Sigma)^-;\Gamma^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1]\!]L : A^-$ by ih
$(\Delta,\Sigma)^-;\Gamma^- \vdash L_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2]\!]L : A^-$ by ih
$(\Delta,\Sigma)^-;\Gamma^- \vdash S_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1']\!]S : B^- > a^-$ by ih
$(\Delta,\Sigma)^-;\Gamma^- \vdash S_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2']\!]S : B^- > a^-$ by ih
$(\Delta,\Sigma)^-;\Gamma^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1,\theta_1']\!]L : A^-$ by lemma 69
$(\Delta,\Sigma)^-;\Gamma^- \vdash L_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2,\theta_2']\!]L : A^-$ by lemma 69
$(\Delta,\Sigma)^-;\Gamma^- \vdash S_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1,\theta_1']\!]S : B^- > a^-$ by lemma 70
$(\Delta,\Sigma)^-;\Gamma^- \vdash S_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2,\theta_2']\!]S : B^- > a^-$ by lemma 70
$(\Delta,\Sigma)^-;\Gamma^- \vdash (L_1;S_1) \longleftrightarrow ([\![\mathsf{id}_{\Delta^-},\theta_1,\theta_1']\!]L; [\![\mathsf{id}_{\Delta^-},\theta_1,\theta_1']\!]S) : a^-$ by rule
$(\Delta,\Sigma)^-;\Gamma^- \vdash (L_1;S_1) \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1,\theta_1']\!](L;S) : a^-$ by modal subst. def
$(\Delta,\Sigma)^-;\Gamma^- \vdash (L_2;S_2) \longleftrightarrow ([\![\mathsf{id}_{\Delta^-},\theta_2,\theta_2']\!]L; [\![\mathsf{id}_{\Delta^-},\theta_2,\theta_2']\!]S) : a^-$ by rule
$(\Delta,\Sigma)^-;\Gamma^- \vdash (L_2;S_2) \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2,\theta_2']\!](L;S) : a^-$ by modal subst. def
$(\Delta,\Sigma) \vdash (\theta_1,\theta_1') : (\Sigma_1,\Sigma_2)$ since θ_1 and θ_1' refer to distinct modal variables
$(\Delta,\Sigma) \vdash (\theta_2,\theta_2') : \Sigma_1,\Sigma_2)$ since θ_2 and θ_2' refer to distinct modal variables

□

2

Recall that we assume that all modal variables have been lowered and are of atomic type.

Theorem 72 (Completeness)

1. *If* $\Delta,\Sigma \vdash \theta_1 : \Sigma'$ *and* $\Delta,\Sigma \vdash \theta_2 : \Sigma'$ *and* θ_1 *and* θ_2 *are incompatible and* $(\Delta,\Sigma)^-;\Gamma^- \vdash L_1 \longleftrightarrow [\![id_\Delta,\theta_1]\!]L : A^-$ *and* $(\Delta,\Sigma)^-;\Gamma^- \vdash L_2 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_2]\!]L : A^-$

[2]Note, that $(\Delta,\Sigma');\Gamma \vdash L : A$ is not quite correct, since there may be type dependencies (in particular if L is an application $R\ L'$, then the type A may contain the term L'). Only $(\Delta,\Sigma);\Gamma \vdash [\![\mathsf{id}_\Delta,\theta_1]\!]L : A$ or $(\Delta,\Sigma);\Gamma \vdash [\![\mathsf{id}_\Delta,\theta_2]\!]L : A$ is true. This follows from the previous soundness theorem, together with soundness of $\longleftrightarrow$ and validity. I am not sure if this matters at all, but it seems surprising.

then there exists a modal substitution θ_1^*, θ_2^*, *and a modal context* Σ^*, *such that* $(\Delta,\Sigma)^-;\Gamma^- \vdash L_1 \sqcup L_2 : A^- \Longrightarrow L/((\Sigma^*)^-,\theta_1^*,\theta_2^*)$ *and* $\theta_1^* \subseteq \theta_1$, $\theta_2^* \subseteq \theta_2$ *and* $\Sigma^* \subseteq \Sigma'$

2. *If* $\Delta,\Sigma \vdash \theta_1 : \Sigma'$ *and* $\Delta,\Sigma \vdash \theta_2 : \Sigma'$ *and* θ_1 *and* θ_2 *are incompatible and* $(\Delta,\Sigma);\Gamma \vdash S_1 \longleftrightarrow [\![id_\Delta,\theta_1]\!]S : A^- > a^-$ *and* $(\Delta,\Sigma);\Gamma \vdash S_2 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_2]\!]S : A^- > a^-$ *then there exists a modal substitution* θ_1^*, θ_2^*, *and a modal context* Σ^*, *such that* $(\Delta,\Sigma)^-;\Gamma^- \vdash S_1 \sqcup\!\!\!\!\_\; S_2 : A^- \Longrightarrow S/(\Sigma^{*-},\theta_1^*,\theta_2^*)$ *and* $\theta_1^* \subseteq \theta_1$, $\theta_2^* \subseteq \theta_2$ *and* $\Sigma^* \subseteq \Sigma'$.

Proof: Simultanous induction on the structure of L and S.

Case $L = u[\pi]$ and $u{::}\Psi{\vdash}a \in \Delta$

$(\Delta,\Sigma)^-;\Gamma^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_\Delta,\theta_1]\!](u[\pi]) : a^-$	by assumption
$(\Delta,\Sigma)^-;\Gamma^- \vdash L_1 \longleftrightarrow u[\pi] : a^-$	by modal subst. def
$(\Delta,\Sigma)^-;\Gamma^- \vdash u[\pi] \longleftrightarrow u[\pi] : a^-$	by inversion
and $L_1 = u[\pi]$	
$(\Delta,\Sigma)^-;\Gamma^- \vdash u[\pi] \sqcup u[\pi] : a^- \Longrightarrow u[\pi]/(\cdot,\cdot,\cdot)$	by rule
$\cdot \subseteq \Sigma'$, $\cdot \subseteq \theta_1$, $\cdot \subseteq \theta_2$	

Case $L = \lambda x.L'$.

$(\Delta,\Sigma)^-;\Gamma^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1]\!](\lambda x.L') : A^- \to B^-$	by assumption
$(\Delta,\Sigma)^-;\Gamma^- \vdash L_1 \longleftrightarrow \lambda x.[\![\mathsf{id}_{\Delta^-},\theta_1]\!]L' : A^- \to B^-$	by modal subst. def
$(\Delta,\Sigma)^-;\Gamma^-,x{:}A^- \vdash L_1' \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1]\!]L' : B^-$	by inversion
and $L_1 = \lambda x.L_1'$	
$(\Delta,\Sigma)^-;\Gamma^- \vdash L_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2]\!](\lambda x.L') : A^- \to B^-$	by assumption
$(\Delta,\Sigma)^-;\Gamma^- \vdash L_2 \longleftrightarrow \lambda x.[\![\mathsf{id}_{\Delta^-},\theta_2]\!]L' : A^- \to B^-$	by modal subst. def
$(\Delta,\Sigma)^-;\Gamma^-,x{:}A^- \vdash L_2' \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2]\!]L' : B^-$	by inversion
and $L_2 = \lambda x.L_2'$	
$(\Delta,\Sigma)^-;\Gamma^-,x{:}A^- \vdash L_1' \sqcup L_2' : B^- \Longrightarrow L'/(\Sigma'^-,\theta_1,\theta_2)$	by ih
$\Sigma' \subseteq \Sigma'$, $\theta_1 \subseteq \theta_1$, $\theta_2 \subseteq \theta_2$	
$(\Delta,\Sigma)^-;\Gamma^- \vdash \lambda x.L_1' \sqcup \lambda x.L_2' : B^- \Longrightarrow \lambda x.L'/(\Sigma'^-,\theta_1,\theta_2)$	by rule

Case $L = i[\mathsf{id}_{\Gamma^-}]$

$(\Delta, \Sigma)^-; \Gamma^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_\Delta, \theta_1]\!](i[\mathsf{id}_{\Gamma^-}]) : a^-$ by assumption
$(\Delta, \Sigma)^-; \Gamma^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_\Delta, \theta_2]\!](i[\mathsf{id}_{\Gamma^-}]) : a^-$ by assumption
$L'/i \in \theta_1$ and $L''/i \in \theta_2$ by assumption
L' and L'' are incompatible by assumption
$(\Delta, \Sigma)^-; \Gamma^- \vdash L_1 \longleftrightarrow L' : a^-$ by modal sub def
$(\Delta, \Sigma)^-; \Gamma^- \vdash L_1 \longleftrightarrow L'' : a^-$ by modal sub def
$L_1 = L'$ and $L_2 = L''$ by LEMMA

Case: $L_1 = u[\pi]$ and $L_2 = L''$
$(\Delta, \Sigma)^-; \Gamma^- \vdash u[\pi] \sqcup L'' : a^- \Longrightarrow i[\mathsf{id}_{\Gamma^-}]/(i{::}\Gamma^- \vdash a^-, u[\pi]/i, L''/i)$ by rule
$i{::}(\Gamma^- \vdash a^-) \subseteq \Sigma'$, $(u[\pi]/i) \subseteq \theta_1$, $(L''/i) \subseteq \theta_2$,

Case: $L_1 = L'$ and $L_2 = u[\pi]$
$(\Delta, \Sigma)^-; \Gamma^- \vdash L' \sqcup u[\pi] : a^- \Longrightarrow i[\mathsf{id}_{\Gamma^-}]/(i{::}\Gamma^- \vdash a^-, L'/i, u[\pi]/i)$ by rule
$(i{::}(\Gamma^- \vdash a^-)) \subseteq \Sigma'$, $(u[\pi]/i) \subseteq \theta_2$, $(L'/i) \subseteq \theta_1$,

Case: $L_1 = H \cdot S_1$ and $L_2 = H_2 \cdot S_2$
$H_1 \cdot S_1$ is incompatible with $H_2 \cdot S_2$ and $H_1 \neq H_2$ by assumption
$(\Delta, \Sigma)^-; \Gamma^- \vdash H_1 \cdot S_1 \sqcup H_2 \cdot S_2 : a^- \Longrightarrow i[\mathsf{id}_{\Gamma^-}]/(i{::}\Gamma^- \vdash a^-, H_1 \cdot S_1/i, H_2 \cdot S_2/i)$
by rule
$(i{::}(\Gamma^- \vdash a^-)) \subseteq \Sigma'$, $(H_1 \cdot S_1/i) \subseteq \theta_1$, $(H_2 \cdot S_2/i) \subseteq \theta_2$,

These are all the cases; other cases cannot happen due to the assumption that $i[\mathsf{id}]$ must be of atomic type and the two substitutions must be incompatible.

$\square$

Theorem 73 (Soundness for msg of substitutions)
If $(\Lambda, \Theta_1) \vdash \rho_1 \sqcup \rho_2 : \Theta_2 \Longrightarrow \rho/(\Theta, \theta_1, \theta_2)$ *and*
$(\Lambda, \Theta_1) \vdash \rho_1 : \Theta_2$ *and* $(\Lambda, \Theta_1) \vdash \rho_2 : \Theta_2$
then $(\Lambda, \Theta_1) \vdash [\![\theta_1]\!]\rho \longleftrightarrow \rho_1 : \Theta_2$ *and* $(\Lambda, \Theta_1) \vdash [\![\theta_2]\!]\rho \longleftrightarrow \rho_2 : \Theta_2$

Proof: Induction on the first derivation.

Case $\mathcal{D} = \dfrac{}{(\Lambda, \Theta_1) \vdash \cdot : \cdot \Longrightarrow \cdot/(\cdot, \cdot, \cdot)}$

$(\Lambda, \Theta_1) \vdash \cdot \longleftrightarrow \cdot : \cdot$ by rule

$(\Lambda, \Theta_1) \vdash \cdot \longleftrightarrow [\![\cdot]\!](\cdot) : \cdot$ modal subst definition

Case $\mathcal{D} = (\Delta, \Sigma_1)^- \vdash (\rho_1, L_1/i) \sqcup (\rho_2, L_2/i) : (\Sigma_2^-, i{::}(\Psi^- \vdash a^-))$
$\Longrightarrow (\rho, L/i)/((\Sigma, \Sigma')^-, (\theta_1, \theta_1'), (\theta_2, \theta_2'))$

$(\Delta, \Sigma_1)^- \vdash \rho_1 \sqcup \rho_2 : \Sigma_2^- \Longrightarrow \rho/(\Sigma^-, \theta_1, \theta_2)$ by inversion

$(\Delta, \Sigma_1)^-; \Psi^- \vdash L_1 \sqcup L_2 : a^- \Longrightarrow L/(\Sigma'^-, \theta_1', \theta_2')$ by inversion

$(\Delta, \Sigma_1) \vdash (\rho_1, L_1/i) : (\Sigma_2, i{::}(\Psi \vdash a))$ by assumption

$(\Delta, \Sigma_1) \vdash \rho_1 : \Sigma_2$ by inversion

$(\Delta, \Sigma_1); \Psi \vdash L_1 : a$

$(\Delta, \Sigma_1) \vdash (\rho_2, L_2/i) : (\Sigma_2, i{::}(\Psi \vdash A))$ by assumption

$(\Delta, \Sigma_1) \vdash \rho_2 : \Sigma_2$ by inversion

$(\Delta, \Sigma_1); \Psi \vdash L_2 : A$

$(\Delta, \Sigma_1)^-; \Psi^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_1']\!]L : A^-$ by lemma 71

$(\Delta, \Sigma_1)^-; \Psi^- \vdash L_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_2']\!]L : A^-$ by lemma 71

$(\Delta, \Sigma_1)^-; \Psi^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_1, \theta_1']\!]L : A^-$ by lemma 70

$(\Delta, \Sigma_1)^-; \Psi^- \vdash L_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_2, \theta_2']\!]L : A^-$ by lemma 70

$(\Delta, \Sigma_1)^- \vdash \rho_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_1]\!]\rho : \Sigma_2^-$ by ih

$(\Delta, \Sigma_1)^- \vdash \rho_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_2]\!]\rho : \Sigma_2^-$ by ih

$(\Delta, \Sigma_1)^- \vdash \rho_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_1, \theta_1']\!]\rho : \Sigma_2^-$ by lemma 69

$(\Delta, \Sigma_1)^- \vdash \rho_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_2, \theta_2']\!]\rho : \Sigma_2^-$ by lemma 69

$(\Delta, \Sigma_1)^- \vdash (\rho_1, L_1/i) \longleftrightarrow ([\![\mathsf{id}_{\Delta^-}, \theta_1, \theta_1']\!]\rho,\ [\![\mathsf{id}_{\Delta^-}, \theta_1, \theta_1']\!]L/i) : \Sigma_2^-, i{::}\Psi^- {\vdash} A^-$ by rule

$(\Delta, \Sigma_1)^- \vdash (\rho_2, L_2/i) \longleftrightarrow ([\![\mathsf{id}_{\Delta^-}, \theta_2, \theta_2']\!]\rho,\ [\![\mathsf{id}_{\Delta^-}, \theta_2, \theta_2']\!]L/i) : \Sigma_2^-, i{::}\Psi^- {\vdash} A^-$ by rule

$(\Delta, \Sigma_1)^- \vdash (\rho_1, L_1/i) \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_1, \theta_1']\!](\rho, L/i) : \Sigma_2^-, i{::}\Psi^- {\vdash} A^-$ by modal subst def.

$(\Delta, \Sigma_1)^- \vdash (\rho_2, L_2/i) \longleftrightarrow [\![\mathsf{id}_{\Delta^-}, \theta_2, \theta_2']\!](\rho, L/i) : \Sigma_2^-, i{::}\Psi^- {\vdash} A^-$ by modal subst def

□

Theorem 74 (Completeness for msg of modal substitutions)
If $(\Delta, \Sigma) \vdash \theta_1 : \Sigma'$ *and* $(\Delta, \Sigma) \vdash \theta_2 : \Sigma'$ *and* θ_1 *and* θ_2 *are incompatible and*

$(\Delta,\Sigma)^- \vdash \rho_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1]\!]\rho : \Sigma_1$ *and* $(\Delta,\Sigma)^- \vdash \rho_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2]\!]\rho : \Sigma_1$ *then* $(\Delta,\Sigma)^- \vdash \rho_1 \sqcup \rho_2 : \Sigma_1 \Longrightarrow \rho/(\Sigma^*,\theta_1^*,\theta_2^*)$ *such that* $\Sigma^* \subseteq \Sigma'$, $\theta_1^* \subseteq \theta_1$, $\theta_2^* \subseteq \theta_2$.

Proof: Induction on the structure of ρ.

Case $\rho = \cdot$

$(\Delta,\Sigma)^- \vdash \rho_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1]\!](\cdot) : \Sigma_1$	by assumption
$\rho_1 = \cdot$ and $\Sigma_1 = \cdot$	by inversion
$(\Delta,\Sigma)^- \vdash \rho_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2]\!](\cdot) : \Sigma_1$	by assumption
$\rho_2 = \cdot$ and $\Sigma_1 = \cdot$	by inversion
$(\Delta,\Sigma)^- \vdash \cdot \sqcup \cdot : \cdot \Longrightarrow \cdot/(\cdot,\cdot,\cdot)$	by rule
$\cdot \subseteq \Sigma_1$, $\cdot \subseteq \theta_1$, $\cdot \subseteq \theta_2$	

Case $\rho = (\rho, L/i)$

$(\Delta,\Sigma)^- \vdash \rho_1' \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1]\!](\rho, L/i) : {\Sigma_1'}^-$	
$(\Delta,\Sigma)^- \vdash \rho_1' \longleftrightarrow ([\![\mathsf{id}_{\Delta^-},\theta_1]\!](\rho),\ [\![\mathsf{id}_{\Delta^-},\theta_1]\!]L/i) : {\Sigma_1'}^-$	by modal subst definition
$\rho_1' = (\rho_1, L_1/i)$ and $\Sigma_1' = \Sigma_1, i{::}\Psi{\vdash}A$	by inversion
$(\Delta,\Sigma)^- \vdash \rho_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1]\!]\rho : \Sigma_1^-$	
$(\Delta,\Sigma)^-;\Psi^- \vdash L_1 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_1]\!]L : A^-$	
$(\Delta,\Sigma)^- \vdash \rho_2' \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2]\!](\rho, L/i) : {\Sigma_1'}^-$	
$(\Delta,\Sigma)^- \vdash \rho_2' \longleftrightarrow ([\![\mathsf{id}_{\Delta^-},\theta_2]\!](\rho),\ [\![\mathsf{id}_{\Delta^-},\theta_2]\!]L/i) : {\Sigma_1'}^-$	by modal subst definition
$\rho_2' = (\rho_2, L_2/i)$ and $\Sigma_1' = \Sigma_1, i{::}\Psi{\vdash}A$	by inversion
$(\Delta,\Sigma)^- \vdash \rho_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2]\!]\rho : \Sigma_1^-$	
$(\Delta,\Sigma)^-;\Psi^- \vdash L_2 \longleftrightarrow [\![\mathsf{id}_{\Delta^-},\theta_2]\!]L : A^-$	
$(\Delta,\Sigma)^-;\Psi^- \vdash L_1 \sqcup L_2 : A^- \Longrightarrow L/(\Sigma^*,\theta_1^*,\theta_2^*)$	by completeness lemma 72
$\Sigma^* \subseteq \Sigma'$, $\theta_1^* \subseteq \theta_1$, $\theta_2^* \subseteq \theta_2$	
$(\Delta,\Sigma)^- \vdash rho_1 \sqcup \rho_2 : \Sigma_1^- \Longrightarrow \rho/(\Sigma^{**},\theta_1^{**},\theta_2^{**})$	by ih
$\Sigma^{**} \subseteq \Sigma'$, $\theta_1^{**} \subseteq \theta_1$, $\theta_2^{**} \subseteq \theta_2$	
$(\Delta,\Sigma)^- \vdash (rho_1, L_1/i) \sqcup (\rho_2, L_2/i) : (\Sigma_1^-, i{::}\Psi^-{\vdash}A^-$	
$\Longrightarrow (\rho, L/i)/((\Sigma^{**},\Sigma^*),\ (\theta_1^{**},\theta_1^*),\ (\theta_2^{**},\theta_2^*))$	by rule
$(\Sigma^{**},\Sigma^*) \subseteq \Sigma'$, $(\theta_1^{**},\theta_1^*) \subseteq \theta_1$, $(\theta_2^{**},\theta_2^*) \subseteq \theta_2$	
due to subset properties and independence	

$\square$

When inserting a substitution ρ_2 into a substitution tree, we need to traverse the index tree and compute at each node the msg. Before we describe the traversal more formally, we give a more formal definition of substitution trees.

$$\begin{array}{lcl} \text{Node N} & ::= & (\Theta \vdash \rho \twoheadrightarrow C) \\ \text{Children C} & ::= & [N, C] \mid \cdot \end{array}$$

A tree is a node N with a modal substitution ρ and a list of children C. In general, we will write $\Lambda \vdash N : \Theta'$ where $N = (\Theta \vdash \rho \twoheadrightarrow C)$ which means that $(\Lambda, \Theta) \vdash \rho : \Theta'$ and all the children N_i in C, $\Lambda \vdash N_i : \Theta$.

To insert a new substitution ρ_2 in to the substitution tree N where $\Lambda \vdash N : \Theta$, we proceed in two steps. First, we inspect all the children N_i of a parent node N, where $N_i = \Theta_i \vdash \rho_i \twoheadrightarrow C_i$ and check if ρ_1 is compatible with ρ_2. This compatibility check has three possible result:

1. $(\Lambda, \Theta_i) \vdash \rho_i \sqcup \rho_2 : \Theta \Longrightarrow \mathsf{id}_\Theta/(\Theta, \rho_i, \rho_2)$:
   This means ρ_i and ρ_2 are not compatible

2. $(\Lambda, \Theta_i) \vdash \rho_i \sqcup \rho_2 : \Theta \Longrightarrow \rho_1/(\Theta_i, \mathsf{id}_{\Theta_i}, \theta_2)$
   This means ρ_2 is an instance of ρ_i and we will continue to insert θ_2 into the children C_i. In this case $[\![\mathsf{id}_\Delta, \theta_2]\!]\rho_2$ is structurally equivalent to ρ_i and we will call ρ_2 *fully compatible* with ρ_i.

3. $(\Lambda, \Theta_i) \vdash \rho_i \sqcup \rho_2 : \Theta \Longrightarrow \rho'/(\Theta'', \theta_1, \theta_2)$
   ρ_i and ρ_2 are compatible, but we need to replace node N_i with a new node $\Theta'' \vdash \rho' \twoheadrightarrow C'$ where C' contains two children, the child node $\Theta_i \vdash \theta_1 \twoheadrightarrow C_i$ and the child node $\Theta_i \vdash \theta_2 \twoheadrightarrow \cdot$. In this case we will call ρ_2 *partially compatible* with ρ_i.

This idea can be formalized by using the following judgment to filter out all children out from C which are compatible with ρ_2. Moreover, we will distinguish between the fully compatible children, which we collect in V, and the partially compatible children, which we collect in S.

$$\Delta^- \vdash C \sqcup \rho_2 : \Sigma^- \Longrightarrow (V, S)$$

Fully compatible children	V	$::=$	$\cdot \mid V, (N, \theta)$
Partially compatible children	S	$::=$	$\cdot \mid S, (N, \Theta{\vdash}\rho, \theta_1, \theta_2)$

ρ_2 is a modal substitution such that $\Delta \vdash \rho_2 : \Sigma$, and for all the children $C_i = (\Sigma_i{\vdash}\rho_i \twoheadrightarrow C')$ in C, we have $\Delta, \Sigma_i \vdash \rho_i : \Sigma$. Then V and S describe all the children from C which are compatible with ρ_2. We distinguish three cases.

$$\frac{\Delta \vdash C \sqcup \rho_2 : \Sigma \Longrightarrow (V, S) \quad \Delta, \Sigma_1 \vdash \rho_1 \sqcup \rho_2 : \Sigma \Longrightarrow \mathsf{id}_\Sigma/(\Sigma, \rho_1, \rho_2)}{\Delta \vdash [(\Sigma_1{\vdash}\rho_1 \twoheadrightarrow C_1), C] \sqcup \rho_2 : \Sigma \Longrightarrow (V, S)} \; NC$$

$$\frac{\Delta \vdash C \sqcup \rho_2 : \Sigma \Longrightarrow (V, S) \quad \Delta, \Sigma_1 \vdash \rho_1 \sqcup \rho_2 : \Sigma \Longrightarrow \rho_1/(\Sigma_1, \mathsf{id}_{\Sigma_1}, \theta_2)}{\Delta \vdash [(\Sigma_1{\vdash}\rho_1 \twoheadrightarrow C_1), C] \sqcup \rho_2 : \Sigma \Longrightarrow ((V\ ,\ (\Sigma_1{\vdash}\rho_1 \twoheadrightarrow C_1)), S)} \; FC$$

$$\frac{\Delta \vdash C \sqcup \rho_2 : \Sigma \Longrightarrow (V, S) \quad \Delta, \Sigma_1 \vdash \rho_1 \sqcup \rho_2 : \Sigma \Longrightarrow \rho/(\Sigma_2, \theta_1, \theta_2)}{\Delta \vdash [(\Sigma_1{\vdash}\rho_1 \twoheadrightarrow C_1), C] \sqcup \rho_2 : \Sigma \Longrightarrow (V, (S\ ,\ ((\Sigma_1{\vdash}\rho_1 \twoheadrightarrow C_1), \Sigma_2{\vdash}\rho, \theta_1, \theta_2))} \; PC$$

The NC rules describes the case where the child C_i is not compatible with ρ_2. Rule FC describes the case where ρ_2 is fully compatible with the child C_i and the rule PC describes the case where ρ_2 is partially compatible with C_i. Now, we are in the position to describe the traversal of the substitution tree.

$$\Delta^- \vdash N \sqcup \rho_2 : \Sigma^- \Longrightarrow N' \quad \text{Insert } \rho_2 \text{ into the substitution tree } N$$

If N is a substitution tree and ρ_2 is not already in the tree, then N' will be the new substitution tree after inserting ρ_2 into N. We write $[N'/N_i]C$ to indicate that the i-th node N_i in the children C is replaced by the new node N'.

Create new leaf

$$\frac{\Delta \vdash C \sqcup \rho_2 : \Sigma' \Longrightarrow (\cdot, \cdot)}{\Delta \vdash (\Sigma{\vdash}\rho \twoheadrightarrow C) \sqcup \rho_2 : \Sigma' \Longrightarrow (\Sigma{\vdash}\rho \twoheadrightarrow (C, (\cdot{\vdash}\rho_2 \twoheadrightarrow \cdot))}$$

Recurse

$$\frac{\Lambda \vdash C \sqcup \rho_2 : \Theta' \Longrightarrow (V, S) \quad N_i \in C \quad (N_i, \theta_2) \in V \quad \Lambda \vdash N \sqcup \theta_2 \Longrightarrow N'}{\Lambda \vdash (\Theta'{\vdash}\rho \twoheadrightarrow C) \sqcup \rho_2 : \Theta \Longrightarrow (\Theta'{\vdash}\rho \twoheadrightarrow [N'/N_i]C}$$

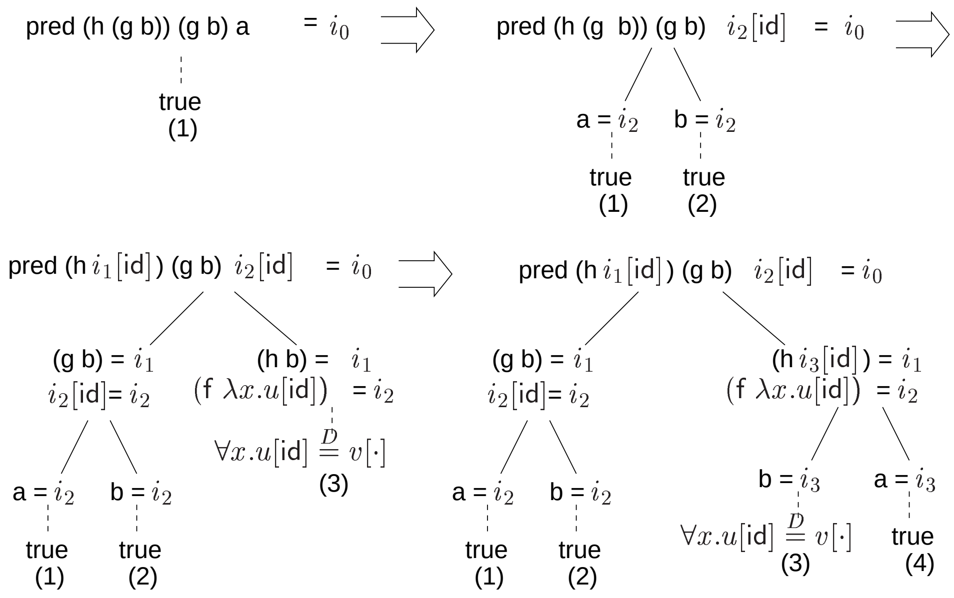
Split current node

$$\frac{\Lambda \vdash C \sqcup \rho_2 : \Theta \Longrightarrow (\cdot, S) \quad N_i \in C \quad N_i = (\Theta_i {\vdash} \rho_i \twoheadrightarrow C_i) \quad (N_i, \Theta^* {\vdash} \rho, \theta_1, \theta_2) \in S}{\Lambda \vdash (\Theta' {\vdash} \rho \twoheadrightarrow C) \sqcup \rho_2 : \Theta \Longrightarrow (\Theta' {\vdash} \rho \twoheadrightarrow [(\Theta^* {\vdash} \rho \twoheadrightarrow ((\Theta_i {\vdash} \theta_1 \twoheadrightarrow C_i), (\Theta_i {\vdash} \theta_2 \twoheadrightarrow \cdot)))/N_i]C)}$$

We illustrate this process by an example. Assume we have the following two modal substitutions:

$$\rho_1 = [\![(g\ a)/i_1,\ f\ (\lambda x.u[\mathsf{id}])/i_2,\ (h\ c)/i_3]\!]$$
$$\rho_2 = [\![(g\ b)/i_1,\ c/i_2,\ (h\ a)/i_3]\!]$$

Then, the most specific linear common generalization of the substitution is computed as follows. We compute the most specific linear generalization for each element. Hence the resulting most specific common substitution ρ of ρ_1 and ρ_2 is $[\![(g\ i_4[\mathsf{id}])/i_1,\ i_2[\mathsf{id}]/i_2,\ (h\ i_5[\mathsf{id}])/i_3]\!]$. In addition, ρ_1 will be split into the most specific common substitution ρ and the substitution $\theta_1 = [\![a/i_4,\ f\ (\lambda x.u[\mathsf{id}])/i_2,\ c/i_5]\!]$, s.t. $[\![\theta_1]\!]\rho = \rho_1$. Similarly ρ_2 will be split into ρ and the substitution $\theta_2 = [\![b/i_4,\ c/i_2,\ a/i_5]\!]$, s.t. $[\![\theta_2]\!]\rho = \rho_2$. Next, we illustrate the process of inserting the terms (1) to (4) from page 128 into a higher-order substitution tree.



In the second step, we can see why it is important to re-use the name of internal modal variables when computing the msg of two terms. Here we need to compute the msg of $\mathsf{pred}\ (\mathsf{h}\ (\mathsf{g}\ \mathsf{b}))\ (\mathsf{g}\ \mathsf{b})\ i_2[\mathsf{id}]$ and $\mathsf{pred}\ (\mathsf{h}\ (\mathsf{h}\ \mathsf{b}))\ (\mathsf{g}\ \mathsf{b})\ (\mathsf{f}\ \lambda x.u[\mathsf{id}])$. The result is the substitution $\theta_1 = [\![(\mathsf{g}\ \mathsf{b})/i_1, i_2[\mathsf{id}]/i_2]\!]$ and $\theta_2 = [\![(\mathsf{h}\ \mathsf{b})/i_1, (\mathsf{f}\ \ \lambda x.u[\mathsf{id}])/i_2]\!]$. Since we chose to re-use the variable i_2 and created a substitution $i_2[\mathsf{id}]/i_2$, we can add a new node with the substitution θ_2 and children $a = i_2$ and $b = i_2$ into the tree without violating the stated invariant or renaming the children.

Finally, we note that we may need to apply weakening before inserting a substitution ρ_2 into the substitution tree N. Consider the first two insertions into the empty substitution tree in the previous example. Since the first two terms inserted, do not refer to any modal variables $\Delta = \cdot$. In other words, if the top node in the substitution tree is well-defined in the empty modal context and $\Delta = \cdot$. In the third step, we are inserting the object $\mathsf{pred}\ (\mathsf{h}\ (\mathsf{h}\ \mathsf{b}))\ (\mathsf{g}\ \mathsf{b})\ (\mathsf{f}\ \lambda x.u[\mathsf{id}])$ which refers to the modal variable u. Hence before inserting the third term, we need to weaken the modal context Δ such that such that $\Delta \vdash N : \Theta$ and $\Delta \vdash \rho_2 : \Theta$.

5.3 Retrieval

In general, we can retrieve all terms from the index which satisfy some property. This property may be finding all terms from the index which unify with a given term U or finding all term V from the index, s.t. a given term U is an instance or variant of V. One advantage of substitution trees is that all these retrieval operations can be implemented with only small changes. A key challenge in the higher-order setting is to design efficient retrieval algorithms which will perform well in practice. In Chapter 3 we presented a unification algorithm for linear higher-order patterns. We specialize this algorithm to check whether the term L_2 is an instances of the term L_1. We treat again internal modal variables differently than global modal variables. In addition, both terms must be linear unlike the algorithm in Chapter 3 where only one of them has to be. The principal judgement are as follows:

$$
\begin{array}{lcll}
(\Delta, \Sigma)^-; \Gamma^- \vdash L_1 & \doteq & L_2 : A^-/(\theta, \rho) & L_2 \text{ is an instance of } L_1 \\
(\Delta, \Sigma)^-; \Gamma^- \vdash S_1 & \doteq & S_2 : A^- > a^-/(\theta, \rho) & S_2 \text{ is an instance of } S_2
\end{array}
$$