

# Term Indexing for the LEO-II Prover

Frank Thei<sup>1</sup>

Christoph Benzmller<sup>2</sup>

<sup>1</sup>FR Informatik, Universitt des Saarlandes, Saarbrcken, Germany

`lime@ags.uni-sb.de`

<sup>2</sup>Computer Laboratory, The University of Cambridge, UK

`chris@ags.uni-sb.de`

## Abstract

We present a new term indexing approach which shall support efficient automated theorem proving in classical higher order logic. Key features of our indexing method are a shared representation of terms, the use of partial syntax trees to speedup logical computations and indexing of subterm occurrences. For the implementation of explicit substitutions, additional support is offered by indexing of bound variable occurrences. A preliminary evaluation of our approach shows some encouraging first results.

## 1 Introduction

Term indexing has become standard in first order theorem proving and is applied in all major systems in this domain [RV02, Sch02, WBH<sup>+</sup>02]. An overview on first order term indexing is given in [RSV01] and [NHRV01] presents an evaluation of different techniques. Comparably few term indexing techniques have been developed and studied for higher order logic. An example is Pientka's work on higher order substitution tree indexing [Pie03].

In this paper we present a new approach to higher order term indexing developed for the higher order resolution prover LEO-II<sup>1</sup>, the successor of LEO [BK98]. Our approach is motivated by work presented in [TSP06], which studies the application of indexing techniques for interfacing between theorem proving and computer algebra.

Pientka's approach is based on substitution tree indexing and relies on unification of linear higher order patterns. While higher order pattern unification is a comparatively high level operation, the approach we present here is based on coordinate and path indexing [Sti89] and thus relies on lower level operations, for example, operations on hashtables. Apart from indexing and retrieval of terms, we particularly want to speedup basic operations such as replacement of (sub-)terms and occurs checks.

---

<sup>1</sup>The LEO-II project at Cambridge University has just started (in October 2006). The project is funded by EPSRC under grant EP/D070511/1.















- Hashtable `occurrences` :  $\mathcal{N} \rightarrow \mathcal{N} \rightarrow PST$  indexes occurrences of subterms (the second key) in a given term (the first key). The indexed value is a PST of the positions where the subterm occurs. If a subterm does not occur, then there is no entry in the hashtable.
- Hashtable `occurs_in` :  $\mathcal{N} \rightarrow \mathcal{N}^*$  is used to index a list of all terms in which a given subterm (the key) occurs.
- Hashtable `occurs_at` :  $pos \rightarrow \mathcal{N} \rightarrow \mathcal{N}^*$  is a hashtable to index all terms in which a given subterm (the second key) occurs at a given position (the first key).

For example, occurrences of symbol  $a$  in the example term  $(= @a)@((\cdot @0)@a)$  are indexed by the following hashtable updates (we assume that  $a$  is represented by term node  $i$  and  $(= @a)@((\cdot @0)@a)$  by term node  $j$ ):

- add  $pst_a$  with first key  $j$  and second key  $i$  in `occurrences`
- add  $j$  with key  $i$  in `occurs_in`
- add  $j$  to the set hashed in `occurs_at` with first key `[func; arg]` and second key  $i$ ; if no such set exists in the hashtable, add the singleton  $\{j\}$
- add  $j$  to the set hashed in `occurs_at` with first key `[arg; arg]` and second key  $i$ ; if no such set exists in the hashtable, add the singleton  $\{j\}$

The basic operations of adding a term to the index take constant time (except for rehashing). The indexing of a term of length  $n$  takes time  $O(n)$ .

### 3.4 Bound Variables

Bound variables play a special role in the term system. To see this, remember our example from the beginning, that is, the term  $\lambda a. \lambda b. ((= b) ((\lambda c. (cb)) a))$  or, with de Bruijn indices,  $\lambda \lambda. ((= x_0) ((\lambda. (x_0 x_1)) x_1))$ . This example shows that two occurrences of the same bound variable may have syntactically different de Bruijn indices and that the de Bruijn indices of occurrences of different variables may be syntactically equal. It is desirable to provide quick access to all variables bound by a given binder to speedup  $\beta$ -reduction and related operations such as raising or lowering of bound variable indices. We will now illustrate our solution to this issue. Remember that indexed terms are always kept in  $\beta\eta$  normal form, hence, normalisation is mandatory after instantiation of existential variables or expansion of defined constants (if the modified terms shall be indexed again).

The syntax tree of our example term in de Bruijn notation is















