- ...

Perfect filtering:

The indexing technique returns exactly those terms satisfying the query.

Imperfect filtering:

The indexing technique returns some superset of the set of all terms satisfying the query.

Retrieval operations must be followed by an additional check, but the index can often be implemented more efficiently.

Frequently: All occurrences of variables are treated as different variables.

Path Indexing

Path indexing:

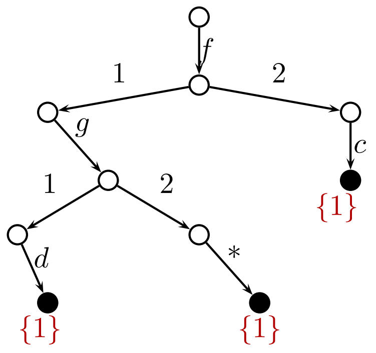
Paths of terms are encoded in a trie ("retrieval tree").

A star $*$ represents arbitrary variables.

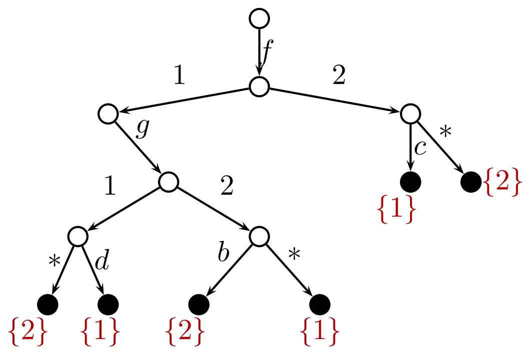
Example: Paths of $f(g(*, b), *)$: $f.1.g.1.*$
$f.1.g.2.b$
$f.2.*$

Each leaf of the trie contains the set of (pointers to) all terms that contain the respective path.

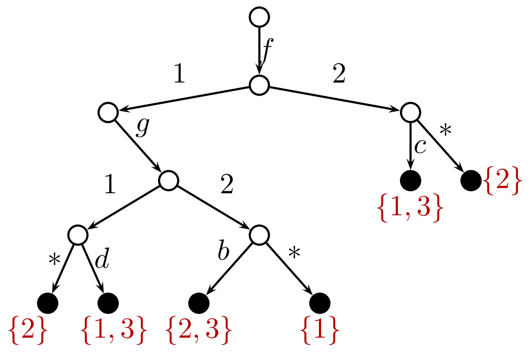
Example: Path index for $\{f(g(d, *), c)\}$



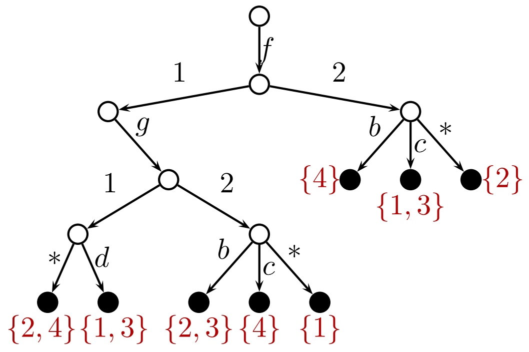
Example: Path index for $\{f(g(d, *), c),\ f(g(*, b), *)\}$



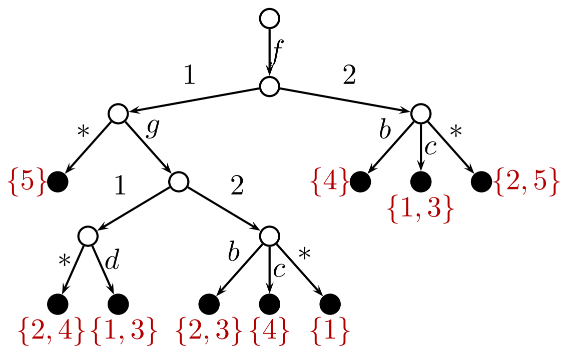
Example: Path index for $\{f(g(d,*),c),\ f(g(*,b),*),\ f(g(d,b),c)\}$



Example: Path index for $\{f(g(d,*),c),\ f(g(*,b),*),\ f(g(d,b),c),\ f(g(*,c),b)\}$



Example: Path index for $\{f(g(d,*),c),\ f(g(*,b),*),\ f(g(d,b),c),\ f(g(*,c),b),\ f(*,*)\}$



Advantages:

Uses little space.

No backtracking for retrieval.

Efficient insertion and deletion.

Good for finding instances.

Disadvantages:

Retrieval requires combining intermediate results for subterms.

Discrimination Trees

Discrimination trees:

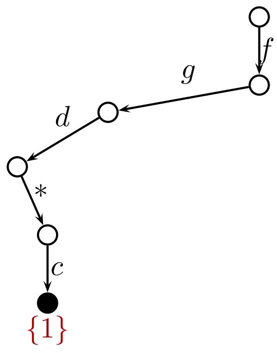
Preorder traversals of terms are encoded in a trie.

A star $*$ represents arbitrary variables.

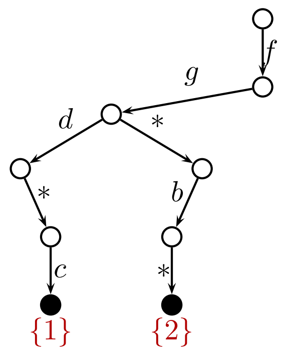
Example: String of $f(g(*, b), *)$: $f.g.*.b.*$

Each leaf of the trie contains (a pointer to) the term that is represented by the path.

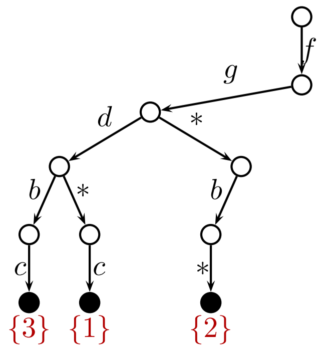
Example: Discrimination tree for $\{f(g(d, *), c)\}$



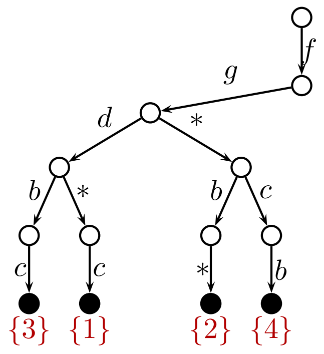
Example: Discrimination tree for $\{f(g(d,*),c),\ f(g(*,b),*)\}$



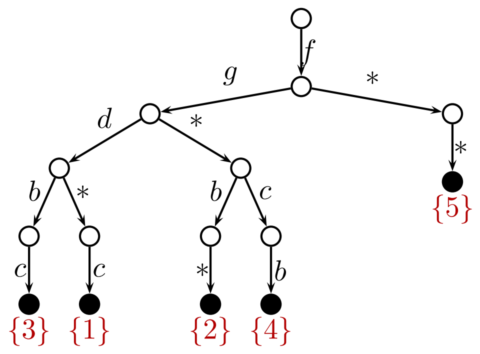
Example: Discrimination tree for $\{f(g(d,*),c),\ f(g(*,b),*),\ f(g(d,b),c)\}$



Example: Discrimination tree for $\{f(g(d,*),c),\ f(g(*,b),*),\ f(g(d,b),c),\ f(g(*,c),b)\}$



Example: Discrimination tree for $\{f(g(d,*),c),\ f(g(*,b),*),\ f(g(d,b),c),\ f(g(*,c),b),$ $f(*,*)\}$



Advantages:

Each leaf yields one term, hence retrieval does not require intersections of intermediate results for subterms.

Good for finding generalizations.

Disadvantages:

Uses more storage than path indexing (due to less sharing).

Uses still more storage, if jump lists are maintained to speed up the search for instances or unifiable terms.

Backtracking required for retrieval.

Literature

Literature:

R. Sekar, I. V. Ramakrishnan, and Andrei Voronkov: Term Indexing, Ch. 26 in Robinson and Voronkov (eds.), *Handbook of Automated Reasoning, Vol. II*, Elsevier, 2001.

Christoph Weidenbach: Combining Superposition, Sorts and Splitting, Ch. 27 in Robinson and Voronkov (eds.), *Handbook of Automated Reasoning, Vol. II*, Elsevier, 2001.

6 Many-Sorted First-Order Logic

Many-Sorted First-order logic

- generalization of first-order logic
- idea is to prohibit ill-defined statements, e.g., $\text{cons}(3, \text{nil}) + 2$
- identical proof theory
- sorts denote subsets of the domain

- variables come with a sort
- functions are declared over the sorts

Many-Sorted Signature

A signature

$$\Sigma_\Upsilon = (\Omega, \Pi, \Upsilon, \upsilon)$$

fixes an alphabet of non-logical symbols, where

- Ω, Π are the sets of function, predicate symbols
- Υ is a set of sort symbols
- υ is a function assigning sorts to function, predicate and variable symbols

Terms, Atoms,Formulae

Well-sorted Terms of sort $S \in \Upsilon$ over Σ_Υ (resp., $\mathrm{T}^S_{\Sigma_\Upsilon}(X)$-terms) are formed according to these syntactic rules:

$$\begin{array}{rcll} s, t, u, v & ::= & x & , x \in X, \upsilon(x) = S \text{ (variable)} \\ & | & f(t_1, ..., t_n) & , f \in \Omega, \mathsf{arity}(f) = n, \upsilon(f) = T_1 \dots T_n S, \\ & & & t_i \in \mathrm{T}^{T_i}_{\Sigma_\Upsilon}(X) \text{ (functional term)} \end{array}$$

By $\mathrm{T}^S_{\Sigma_\Upsilon}$ we denote the set of Σ_Υ-ground terms of sort S, $\mathrm{T}_{\Sigma_\Upsilon}(X) = \bigcup_{S \in \Upsilon} \mathrm{T}^S_{\Sigma_\Upsilon}(X)$.

If $P \in \Pi$, $t_i \in \mathrm{T}^{T_i}_{\Sigma_\Upsilon}(X)$, $\upsilon(P) = T_1 \dots T_n$ then $P(t_1, ..., t_n)$ is an atom. For any $t, s \in \mathrm{T}^S_{\Sigma_\Upsilon}(X)$, $s \approx t$ is an atom.

Formulae are build as for standard (unsorted) first-order logic.

For substitions we additionally require that if $x\sigma = t$ then $t \in \mathrm{T}^{\upsilon(x)}_{\Sigma_\Upsilon}(X)$ and call it *well-sorted*. Note that application of the standard unification algorithms to any two terms of the same sort yields a well-sorted unifier (if there exists a unifier at all).