Global Storing Mechanisms for Tabled Evaluation

Jorge Costa and Ricardo Rocha
DCC-FC & CRACS, University of Porto, Portugal

Introduction

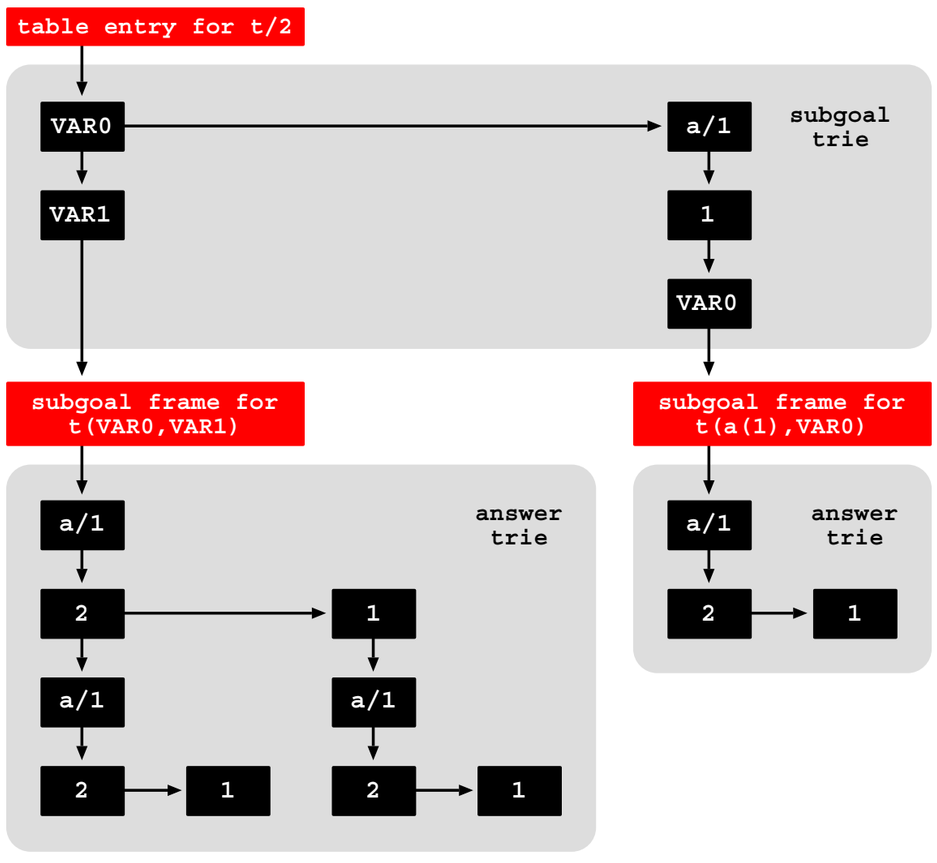
Arguably, the most successful data structure for tabling is tries. However, while tries are very efficient for variant based tabled evaluation, they are limited in their ability to recognize and represent repeated answers for different calls.
We propose a new design for the table space where all tabled subgoal calls and tabled answers are stored in a common global trie instead of being spread over several different trie data structures. Our goal is to share data that is structurally equal, thus saving memory by reducing redundancy in term representation.

Table Space

A trie is a tree structure where each different path through the trie data units, the trie nodes, corresponds to a term. Each root-to-leaf path represents a term described by the tokens labelling the nodes traversed. Two terms with common prefixes will branch off from each other at the first distinguishing token.
Internally, the trie nodes are 4-field data structures. One field stores the node's token, one second field stores a pointer to the node's first child, a third field stores a pointer to the node's parent and a fourth field stores a pointer to the node's next sibling. Each node's outgoing transitions may be determined by following the child pointer to the first child node and, from there, continuing through the list of sibling pointers.
The YapTab tabling system implements tables using two levels of tries - one for subgoal calls, the other for computed answers. More specifically, the table space of YapTab is organized in the following way:

- each tabled predicate has a table entry data structure assigned to it, acting as the entry point for the predicate's subgoal trie.
- each different subgoal call is represented as a unique path in the subgoal trie, starting at the predicate's table entry and ending in a subgoal frame data structure, with the argument terms being stored within the path's nodes.
- the subgoal frame data structure acts as an entry point to the answer trie.
- each different subgoal answer is represented as a unique path in the answer trie. To increase performance, answer trie paths enforce the substitution factoring mechanism and hold just the substitution terms for the free variables which exist in the argument terms.
- the subgoal frame has internal pointers to the first and last answer on the trie and the leaf's child pointer of answers are used to point to the next available answer, a feature that enables answer recovery in insertion time order. Answers are loaded by traversing the answer trie nodes bottom-up.

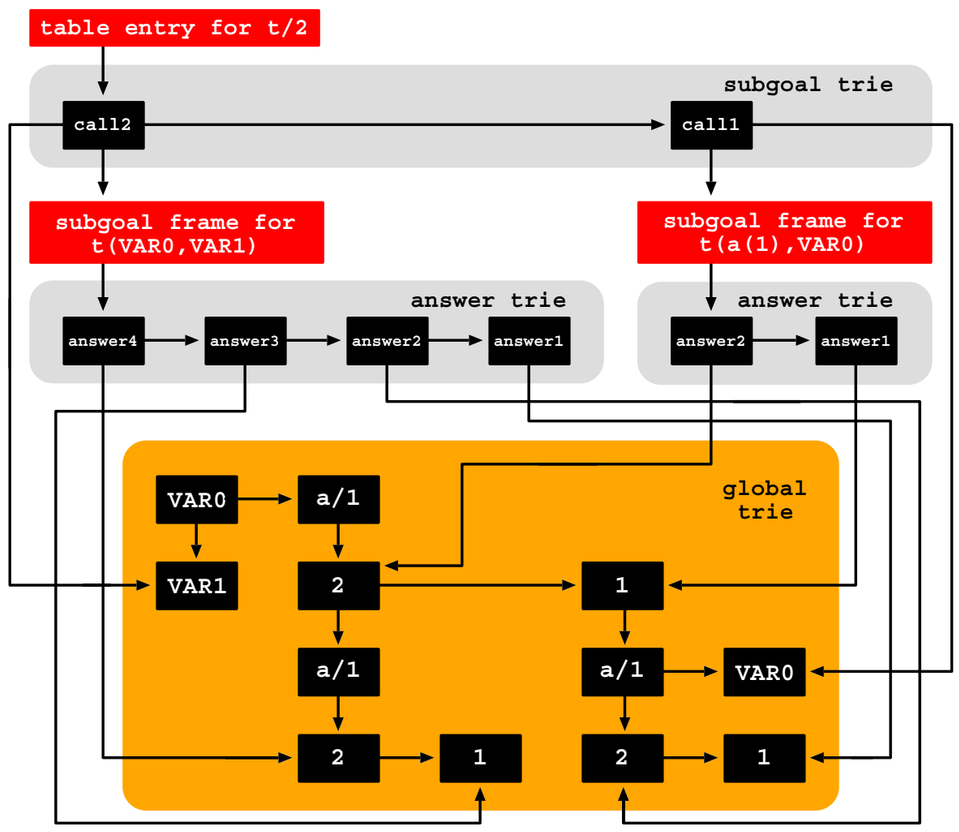
```
:- table t/2.
t(X,Y) :- term(X), term(Y).

term(a(1)).     term(a(2)).
```



YapTab's original table organization

Global Trie

In the new design, all tabled subgoal calls and answers are stored in a common global trie (GT) instead of being spread over several different trie data structures. The GT data structure still is a tree structure where each different path through the trie nodes corresponds to a subgoal call and/or answer. However, here a path can end at any internal trie node and not necessarily at a leaf trie node.
The original subgoal and answer trie data structures are now represented by a unique level of trie nodes that point to the corresponding paths in the GT. For the subgoal tries, each node now represents a different subgoal call where the node's token is the pointer to path in the GT that represents the argument terms for the subgoal call. The organization used in the subgoal tries to maintain the list of sibling nodes and to access the corresponding subgoal frames remains unaltered. For the answer tries, each node now represents a different subgoal answer where the node's token is the pointer to the path in the GT that represents the substitution terms for the free variables which exist in the argument terms. The organization used in the answer tries to maintain the list of sibling nodes and to enable answer recovery in insertion order remains unaltered. With this organization, answers are now loaded by following the pointer in the node's token and then by traversing bottom-up the corresponding nodes in the GT.



YapTab's new table organization

Experimental Results

To evaluate the impact of our proposal, we have defined a tabled predicate t/5 that stores in the table space terms defined by term/1 facts, and then we called it recursively with all combinations of one and two free variables in the arguments. We experimented with 8 different kinds of 500 term/1 facts: integers, atoms and functor terms of arity 1 to 6. The environment for our experiments was an Intel(R) Core(TM)2 Quad 2.66 GHz with 4 GBytes of main memory and running the Linux kernel 2.6.24.7 with YapTab 5.1.4.

Terms	YapTab			YapTab+GT / YapTab		
	Mem (KBytes)	Store (ms)	Load (ms)	Mem	Store	Load
500 ints	49,074	238	88	1.08	1.29	1.05
500 atoms	49,074	256	88	1.08	1.18	1.05
500 f/1	49,172	336	176	1.07	1.33	**0.77**
500 f/2	98,147	430	190	**0.58**	1.16	**0.82**
500 f/3	147,122	554	220	**0.41**	1.04	**0.80**
500 f/4	196,097	596	210	**0.33**	1.07	**0.94**
500 f/5	245,072	676	258	**0.28**	1.00	**0.84**
500 f/6	294,047	796	290	**0.25**	1.01	**0.83**

This work has been partially supported by the research projects STAMPA (PTDC/EIA/67738/2006) and JEDI (PTDC/EIA/66924/2006) and by Fundação para a Ciência e Tecnologia.