Global Storing Mechanisms for Tabled Evaluation

Jorge Costa and Ricardo Rocha
DCC-FC & CRACS
University of Porto, Portugal
c0607002@alunos.dcc.fc.up.pt *ricroc@dcc.fc.up.pt*

ICLP 2008, Udine, Italy, December 2008

Motivation

- The performance of tabling systems largely depends on the implementation of the table space. Arguably, the most successful data structure for tabling is **tries**.

Motivation

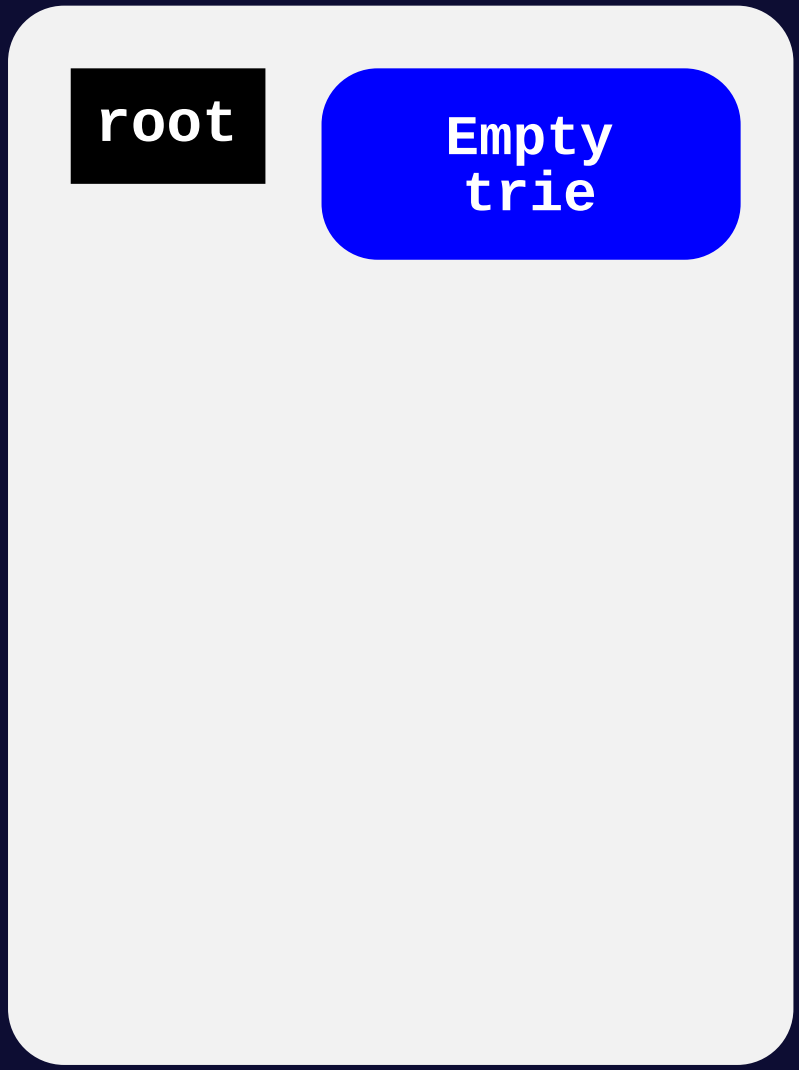
- The performance of tabling systems largely depends on the implementation of the table space. Arguably, the most successful data structure for tabling is **tries**.
- However, while tries are efficient for variant based tabled evaluation, they are **limited in their ability to recognize and represent repeated answers for different calls**.

Motivation

- The performance of tabling systems largely depends on the implementation of the table space. Arguably, the most successful data structure for tabling is **tries**.
- However, while tries are efficient for variant based tabled evaluation, they are **limited in their ability to recognize and represent repeated answers for different calls**.
- In this work, 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.

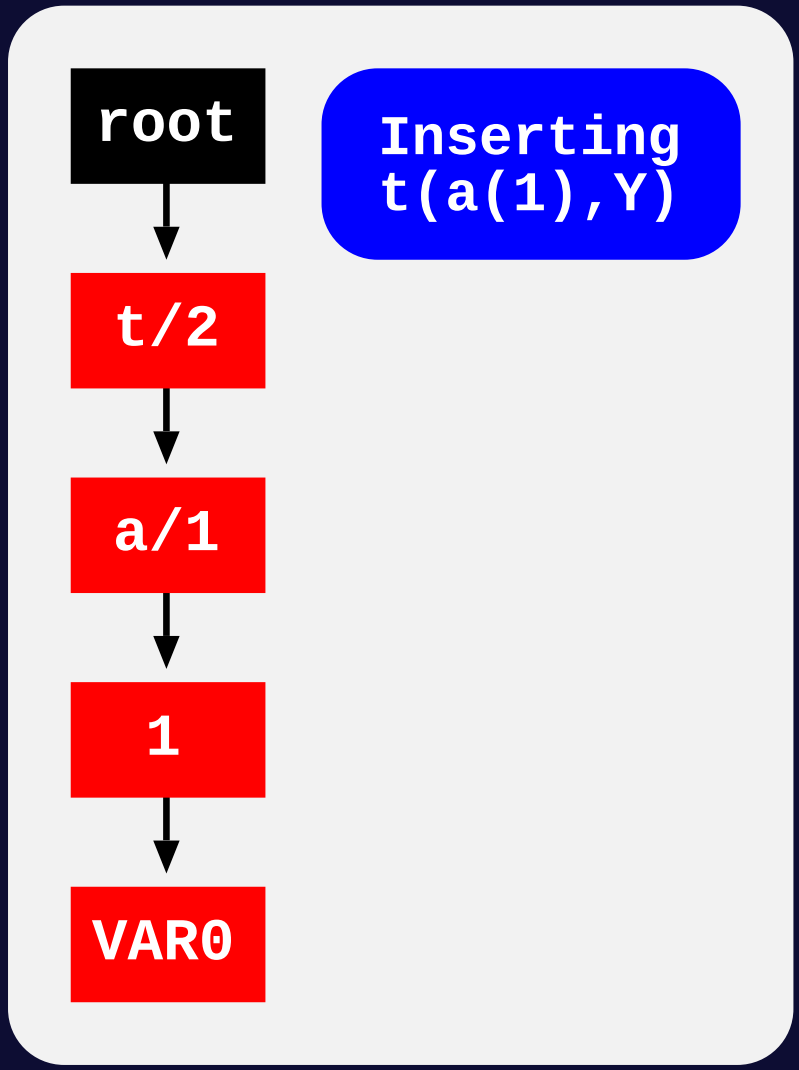
Using Tries to Represent Terms

- Tries are trees in which common prefixes are represented only once.



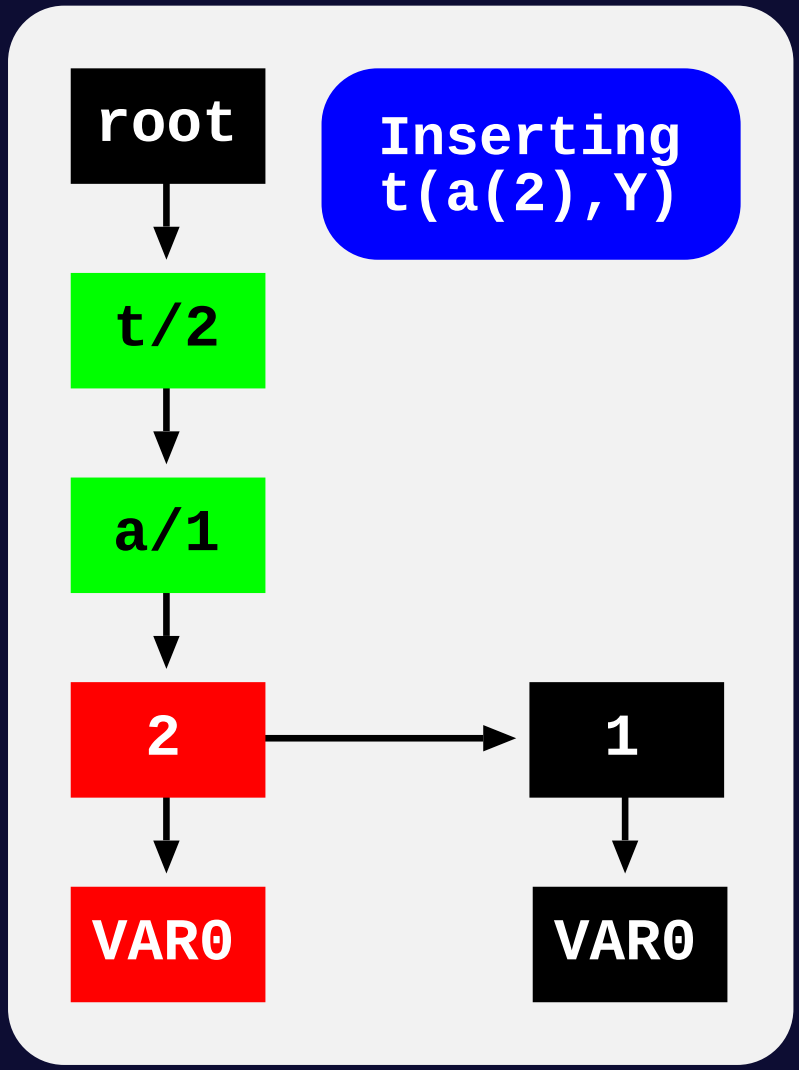
Using Tries to Represent Terms

- Tries are trees in which common prefixes are represented only once.
  - The entry point is called the root node, internal nodes represent symbols in terms and leaf nodes specify completed terms.



Using Tries to Represent Terms

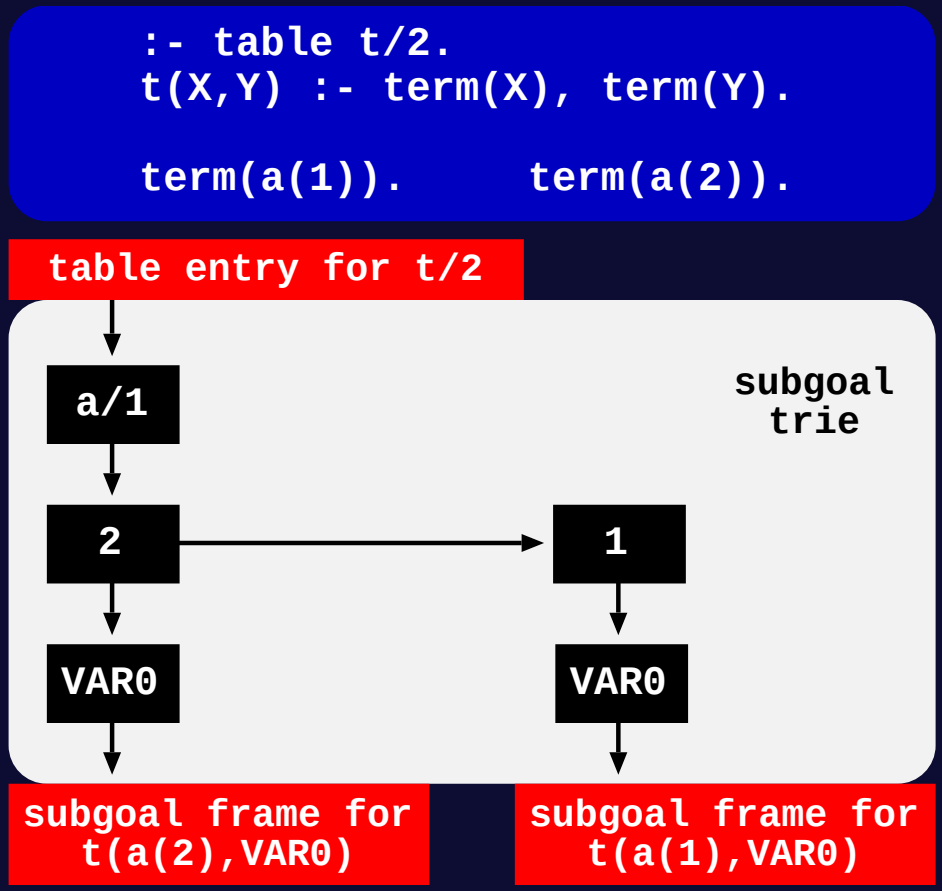
- Tries are trees in which common prefixes are represented only once.
  - The entry point is called the root node, internal nodes represent symbols in terms and leaf nodes specify completed terms.
  - Each different path through the nodes in the trie corresponds to a term. Terms with common prefixes branch off from each other at the first distinguishing symbol.



Using Tries to Organise the Table Space

Subgoal Trie Structure

- Stores the tabled subgoal calls.
- Starts at a table entry and ends with subgoal frames.
- A subgoal frame is the entry point for the subgoal answers.



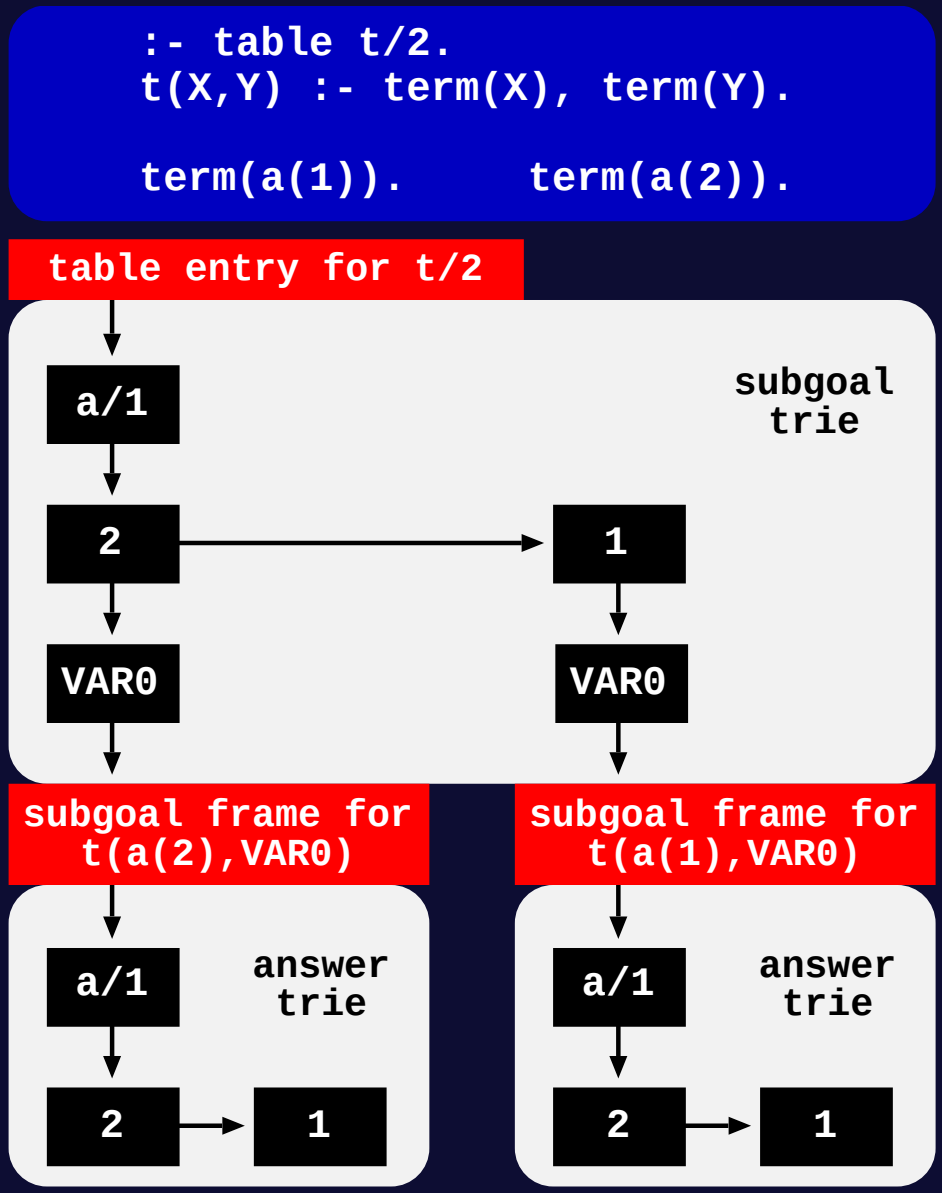
Using Tries to Organise the Table Space

Subgoal Trie Structure

- Stores the tabled subgoal calls.
- Starts at a table entry and ends with subgoal frames.
- A subgoal frame is the entry point for the subgoal answers.

Answer Trie Structure

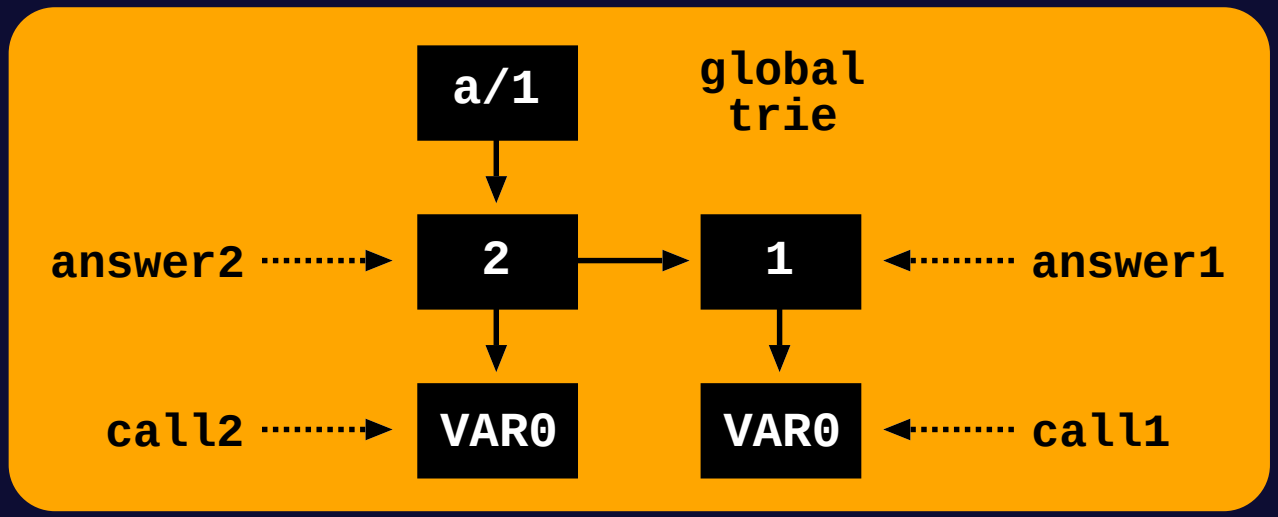
- Stores the subgoal answers.
- Answer tries hold just the substitution terms for the free variables which exist in the argument terms of the corresponding subgoal call.



Commom Global Trie

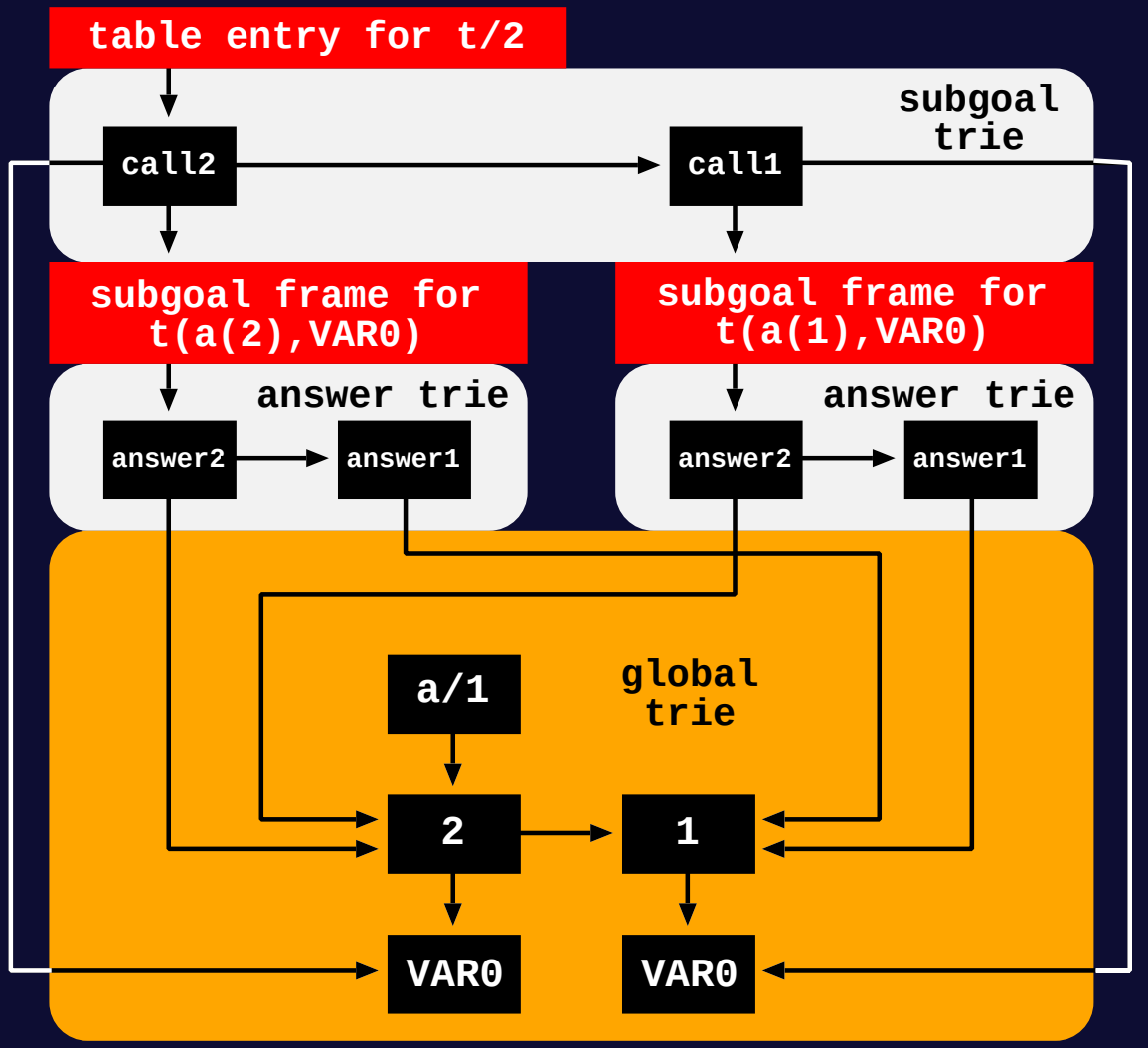
Global Trie Structure

- All tabled subgoal calls and tabled 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.



Commom Global Trie

- The original subgoal trie 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 is a pointer to the GT's path representing the subgoal call.
- For the answer tries, each node is a pointer to the GT's path representing the answer.



Experimental Results in the YapTab System

Terms	YapTab			YapTab+GT / YapTab		
	Mem	Store	Load	Mem	Store	Load
500 ints	49074	238	88	1.08	1.29	1.05
500 atoms	49074	256	88	1.08	1.18	1.05
500 f/1	49172	336	176	1.07	1.33	**0.77**
500 f/2	98147	430	190	**0.58**	1.16	**0.82**
500 f/3	147122	554	220	**0.41**	1.04	**0.80**
500 f/4	196097	596	210	**0.33**	1.07	**0.94**
500 f/5	245072	676	258	**0.28**	1.00	**0.84**
500 f/6	294047	796	290	**0.25**	1.01	**0.83**
Average				**0.64**	**1.14**	**0.89**

Memory usage in KBytes and store/load times in milliseconds for a t/5 tabled predicate that simply stores in the table space terms defined by `term/1` facts, called recursively with all combinations of one and two free variables in the arguments.

Conclusions and Further Work

- We have presented a new design for the table space organization 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 reduce redundancy in term representation, thus saving memory by sharing data that is structurally equal.
- Our preliminary experiments showed very significant reductions on memory usage.
- As further work we intend to study how alternative designs for the table space organization can further reduce redundancy in term representation.