Speculative Computations in Or-Parallel Tabled Logic Programs

Ricardo Rocha Fernando Silva
{*ricroc,fds*}*@ncc.up.pt*
DCC-FC & LIACC, University of Porto, Portugal

Vítor Santos Costa
vitor@cos.ufrj.br
COPPE Systems & LIACC, University of Rio de Janeiro, Brazil

Tabling in Logic Programming

- **Tabling** is an implementation technique where answers for subcomputations are stored and then reused when a repeated computation appears.
  - Tabled subgoals are evaluated by storing their answers in an appropriate data space, called the **table space**.
  - Variant calls to tabled subgoals are resolved by **consuming** the answers already stored in the table instead of being re-evaluated against the program clauses.

- Tabling has proven to be particularly effective in logic (**Prolog**) programs:
  - Avoids recomputation, thus reducing the search space.
  - Avoids infinite loops, thus ensuring termination for a wider class of programs.

Parallel Tabling

- Tabled programs show great potential for **parallel execution**:
  - Programs still need to exploit alternatives for solving goals.
  - Programs often perform search.
  - Programs do not depend on answer ordering.
  - Programs with long running times.

Parallel Tabling

- Tabled programs show great potential for **parallel execution**:
  - Programs still need to exploit alternatives for solving goals.
  - Programs often perform search.
  - Programs do not depend on answer ordering.
  - Programs with long running times.

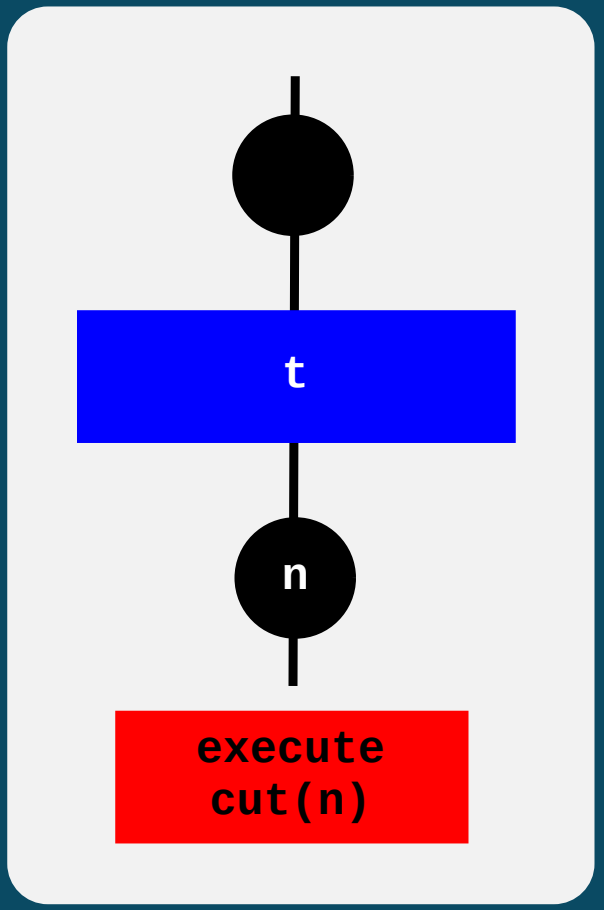
- In previous work we have developed **OPTYap**:
  - The first parallel tabling system for logic programming.
  - Extends the Yap Prolog system to exploit implicit or-parallelism from tabled logic programs.
  - The or-parallel component is based on the environment copying model and the tabling component is mainly based on XSB's SLG-WAM engine [ICLP'01].

Speculative Computations

- The presence of pruning operators, such as the **cut operator**, during parallel execution introduces the problem of speculative computations.
  - Alternatives picked for parallel execution, may later be pruned away by a cut.
  - Earlier execution of such computations results in wasted effort when compared to sequential execution.
  - However, in order to be efficient, most parallel systems allow speculative computations.

Tabling and Pruning

- We can consider two types of cut operations in a tabling environment:
  - **Inner cuts** - cuts that do not prune tabled predicates.



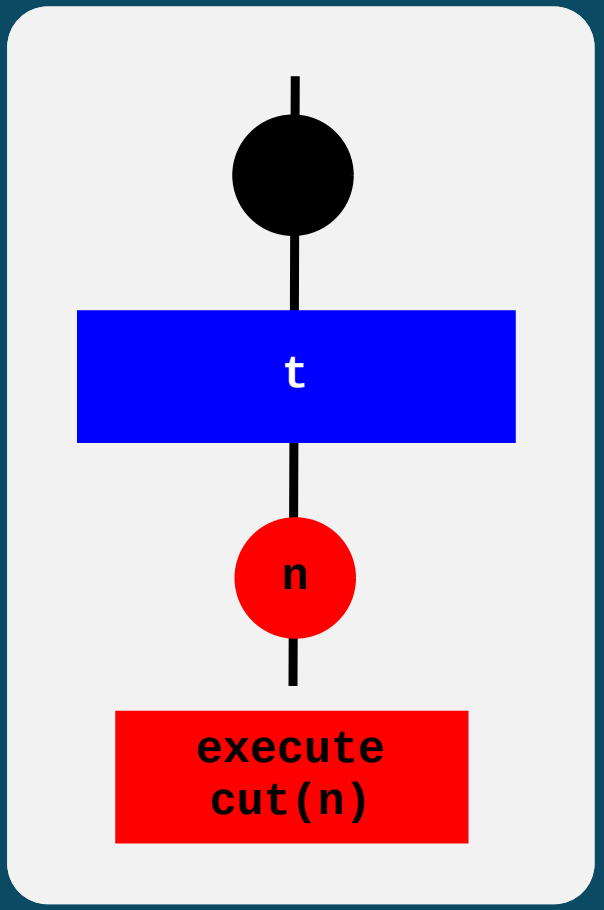

```
:- table t/0.

t :- ... , n, ...
t :- ...

n :- ..., !, ...
n :- ...
```

Tabling and Pruning

- We can consider two types of cut operations in a tabling environment:
  - **Inner cuts** - cuts that do not prune tabled predicates.



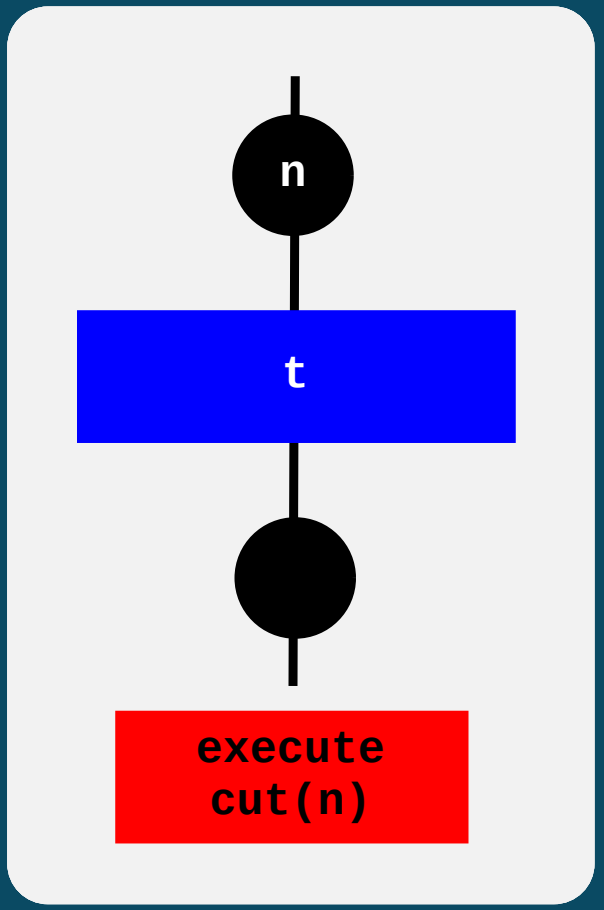

```
:- table t/0.

t :- ... , n, ...
t :- ...

n :- ..., !, ...
n :- ...
```

Tabling and Pruning

- We can consider two types of cut operations in a tabling environment:
  - **Inner cuts** - cuts that do not prune tabled predicates.
  - **Outer cuts** - cuts that prune tabled predicates.



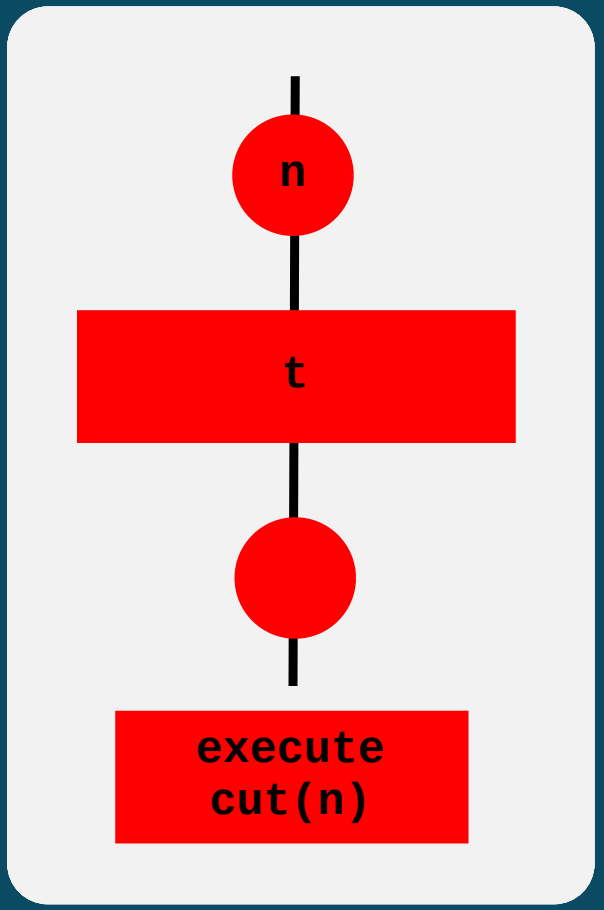

```
:- table t/0.

n :- ..., t, ..., !, ...
n :- ...

t :- ...
t :- ...
```

Tabling and Pruning

- We can consider two types of cut operations in a tabling environment:
  - **Inner cuts** - cuts that do not prune tabled predicates.
  - **Outer cuts** - cuts that prune tabled predicates.




```
:- table t/0.

n :- ..., t, ..., !, ...
n :- ...

t :- ...
t :- ...
```

Tabling and Pruning

- Inner cuts are easily supported in sequential systems.

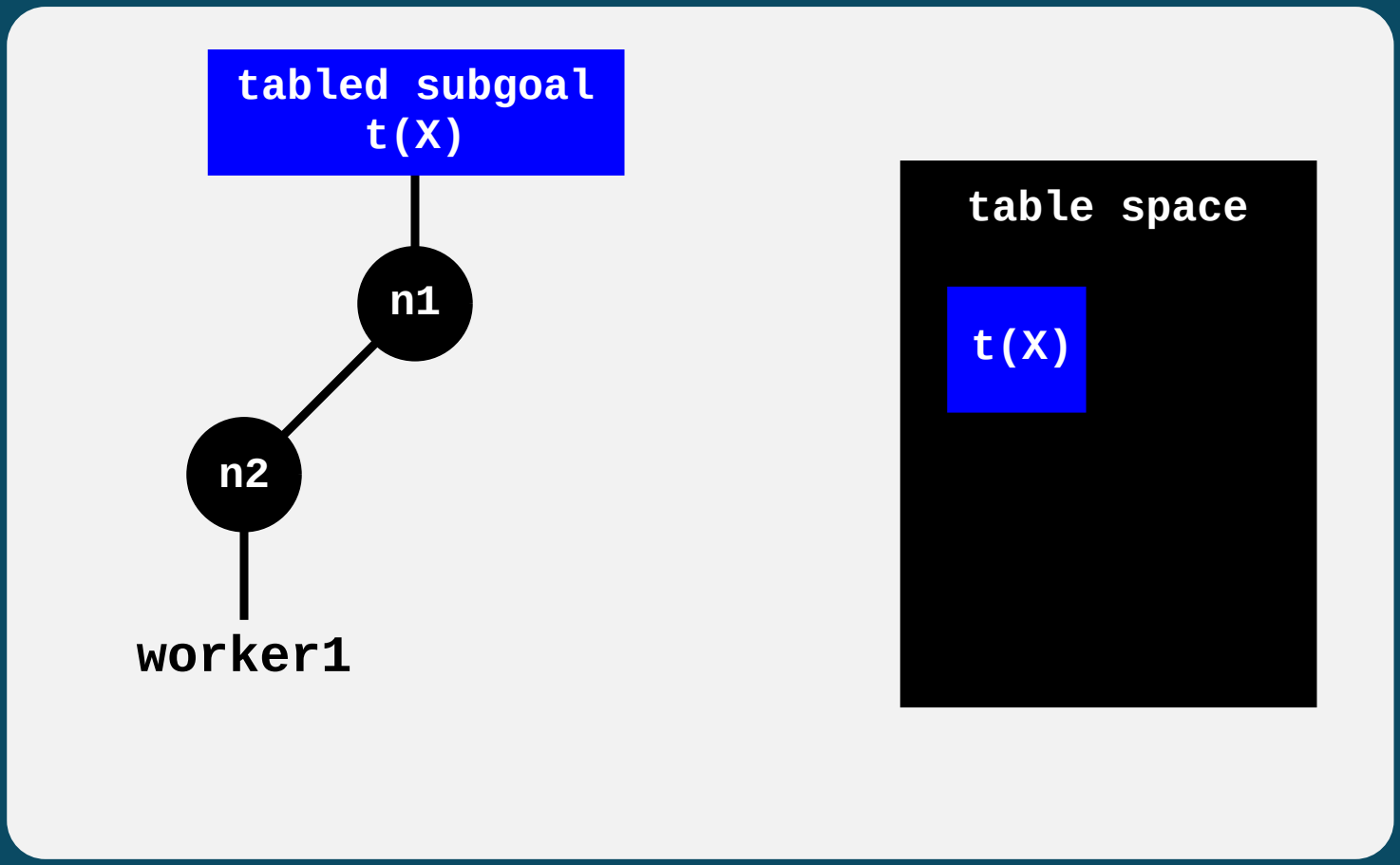
- Outer cuts present major difficulties both in terms of semantics and of implementation. Tabling intrinsically changes the left-to-right semantics of Prolog.
- Current applications do require support for inner pruning. In contrast, outer pruning is not widely used. A complete design for outer cut operations is still an open problem.

Tabling and Pruning

- Inner cuts are easily supported in sequential systems.
- Outer cuts present major difficulties both in terms of semantics and of implementation. Tabling intrinsically changes the left-to-right semantics of Prolog.
- Current applications do require support for inner pruning. In contrast, outer pruning is not widely used. A complete design for outer cut operations is still an open problem.
- In this work we address the problem of how to support **inner pruning in or-parallel tabling**.

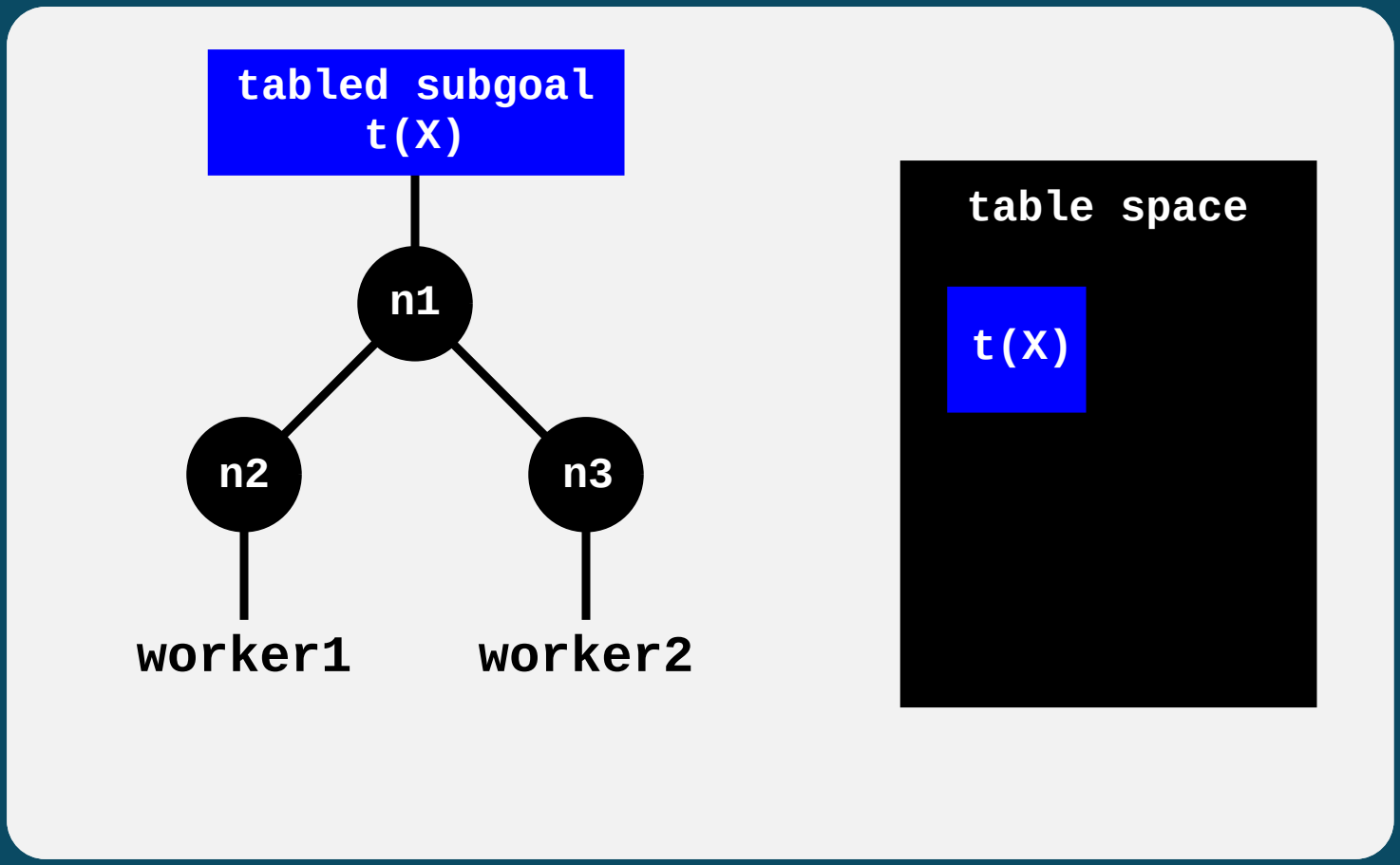
Tabling and Speculative Computations

➤ **Problem I** - Speculative answers found for tabled subgoals may be pruned.



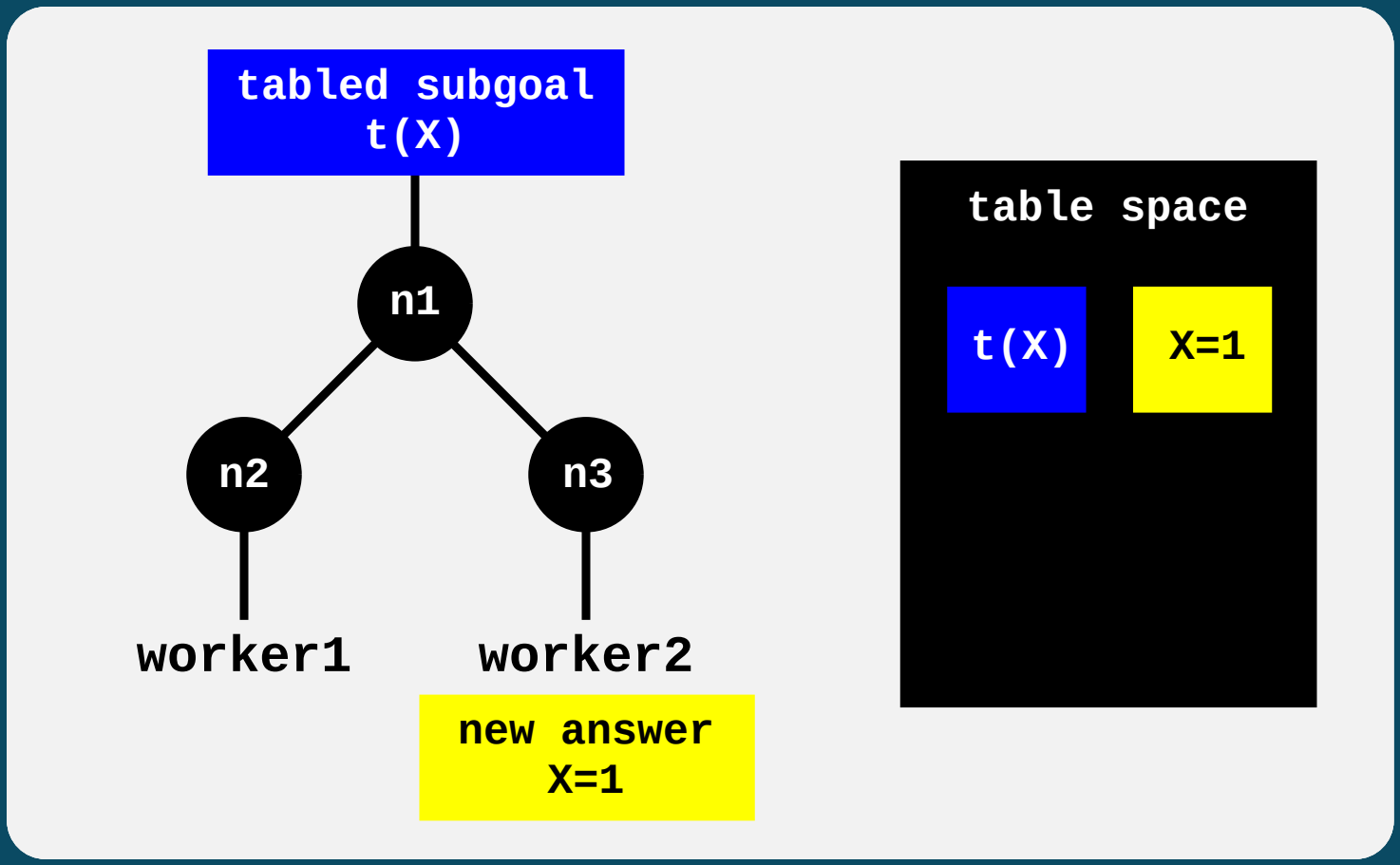
Tabling and Speculative Computations

➤ **Problem I** - Speculative answers found for tabled subgoals may be pruned.



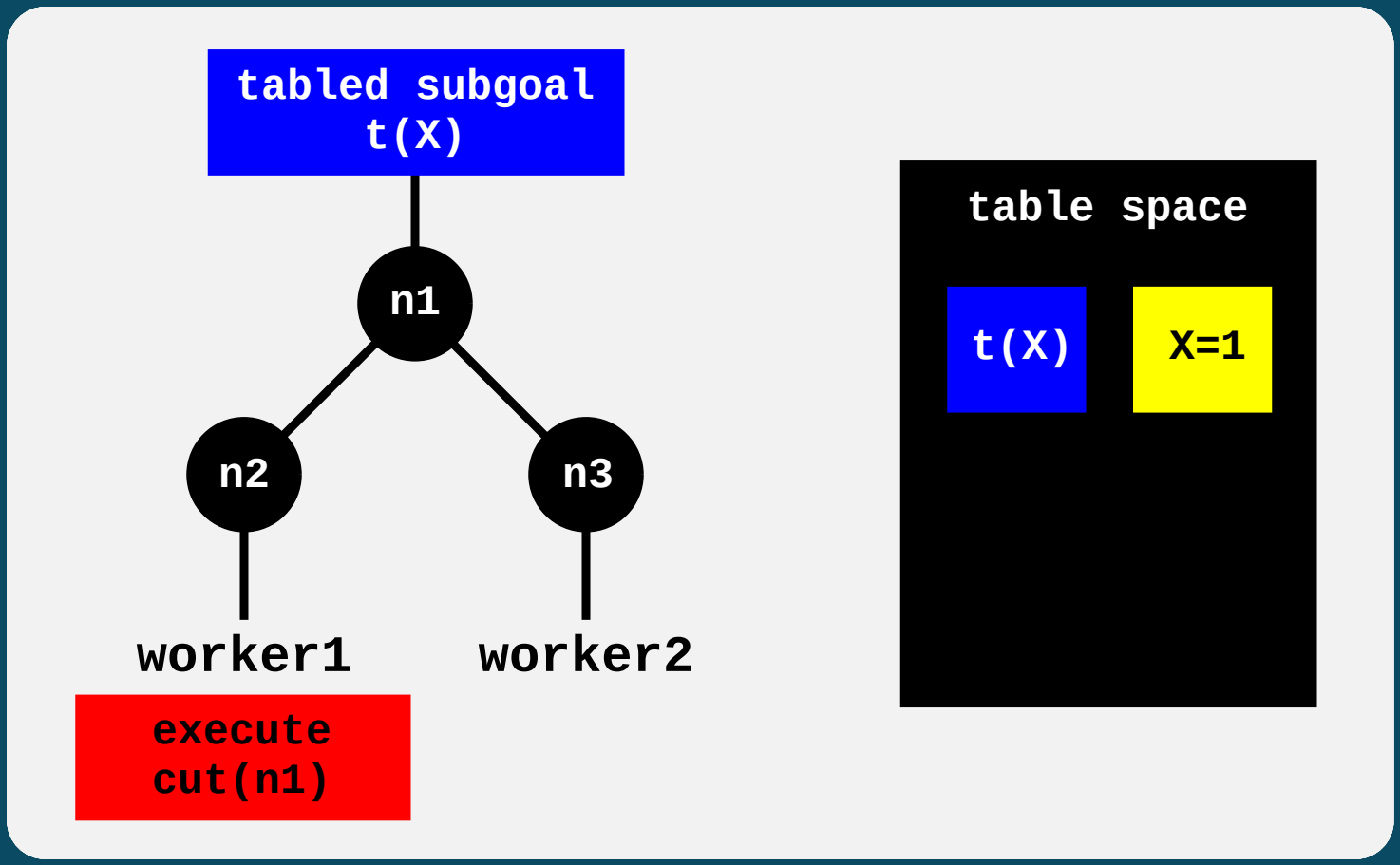
Tabling and Speculative Computations

➤ **Problem I** - Speculative answers found for tabled subgoals may be pruned.



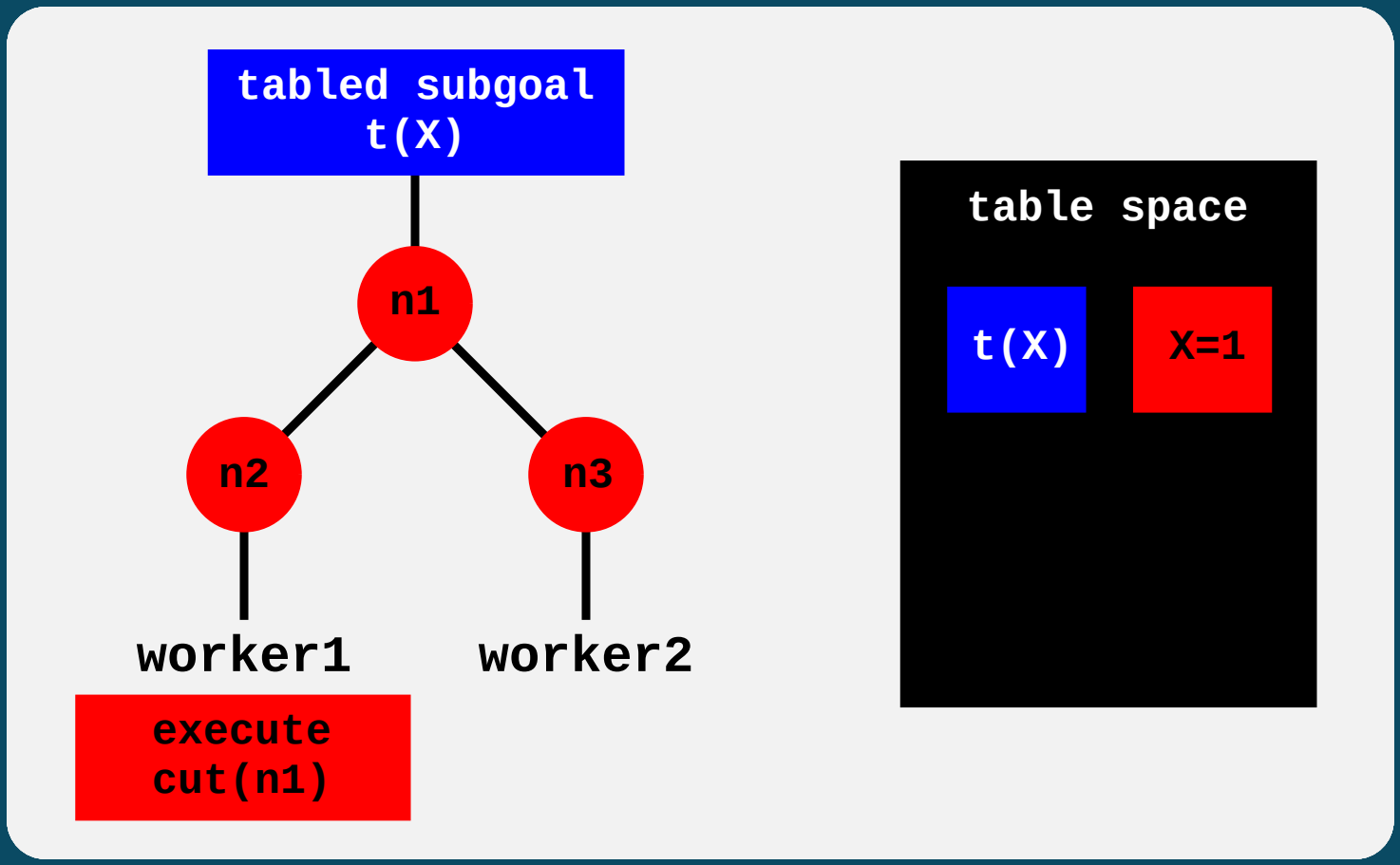
Tabling and Speculative Computations

➤ **Problem I** - Speculative answers found for tabled subgoals may be pruned.



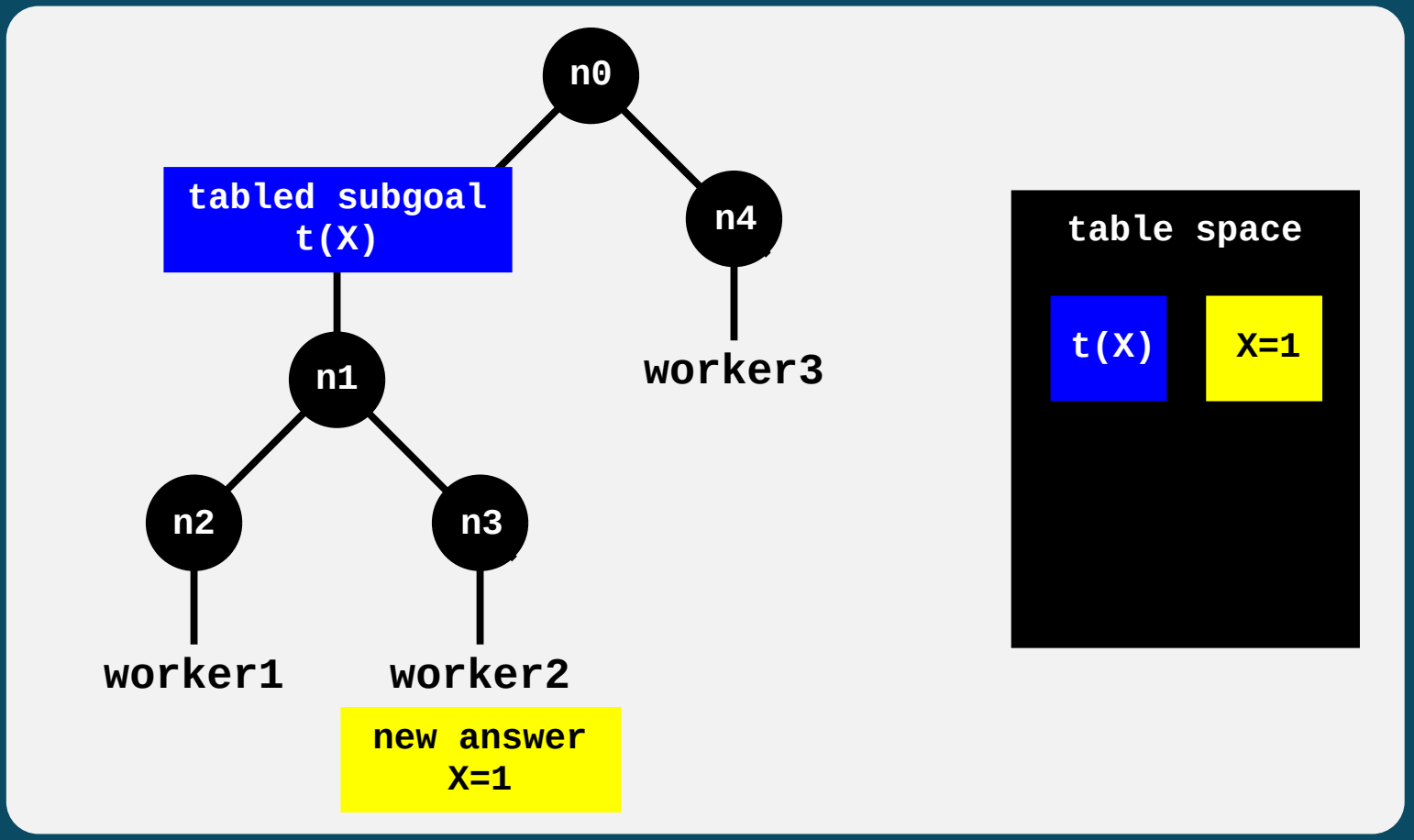
Tabling and Speculative Computations

➤ **Problem I** - Speculative answers found for tabled subgoals may be pruned.



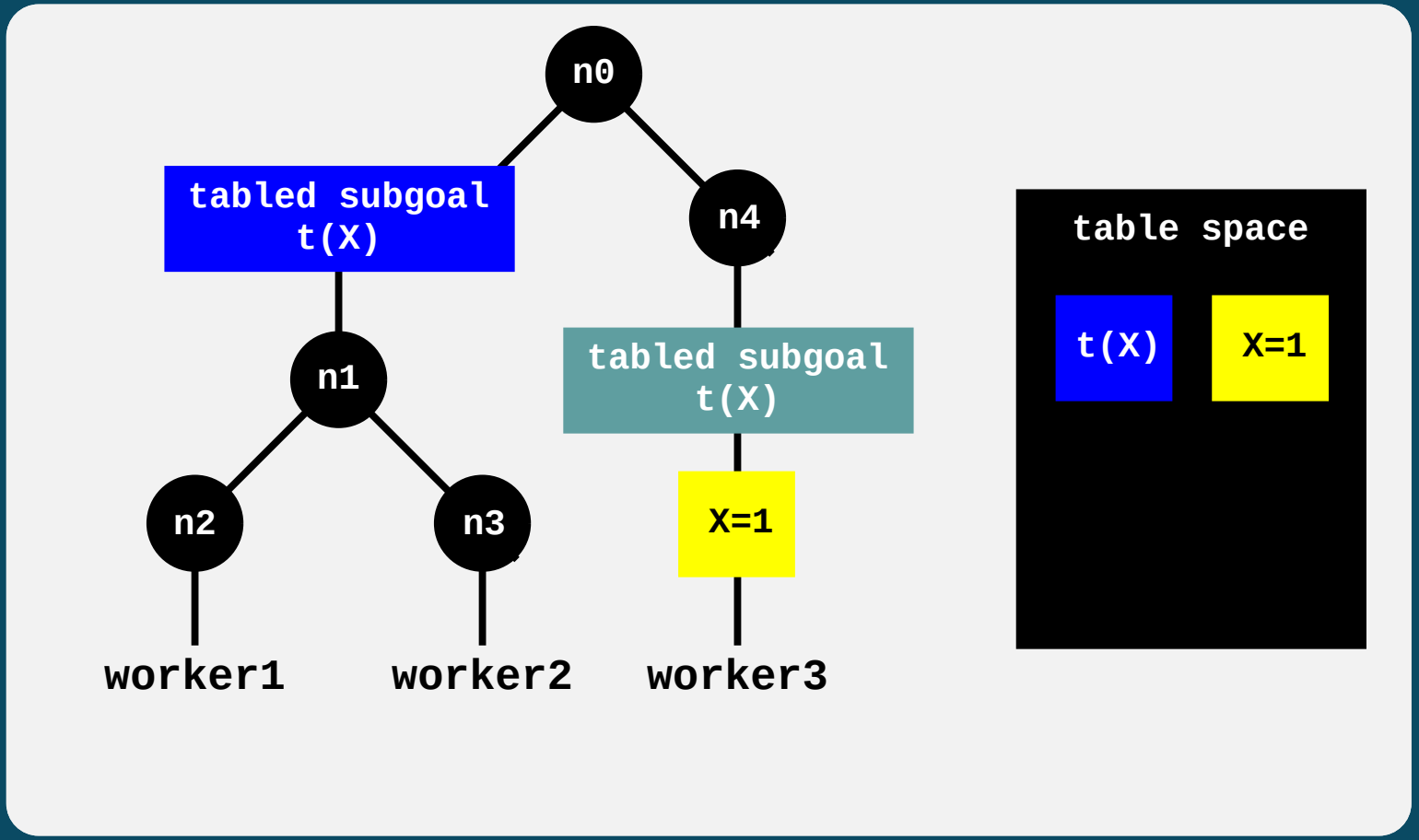
Tabling and Speculative Computations

- **Problem II** - Speculative answers may generate further speculative computations.



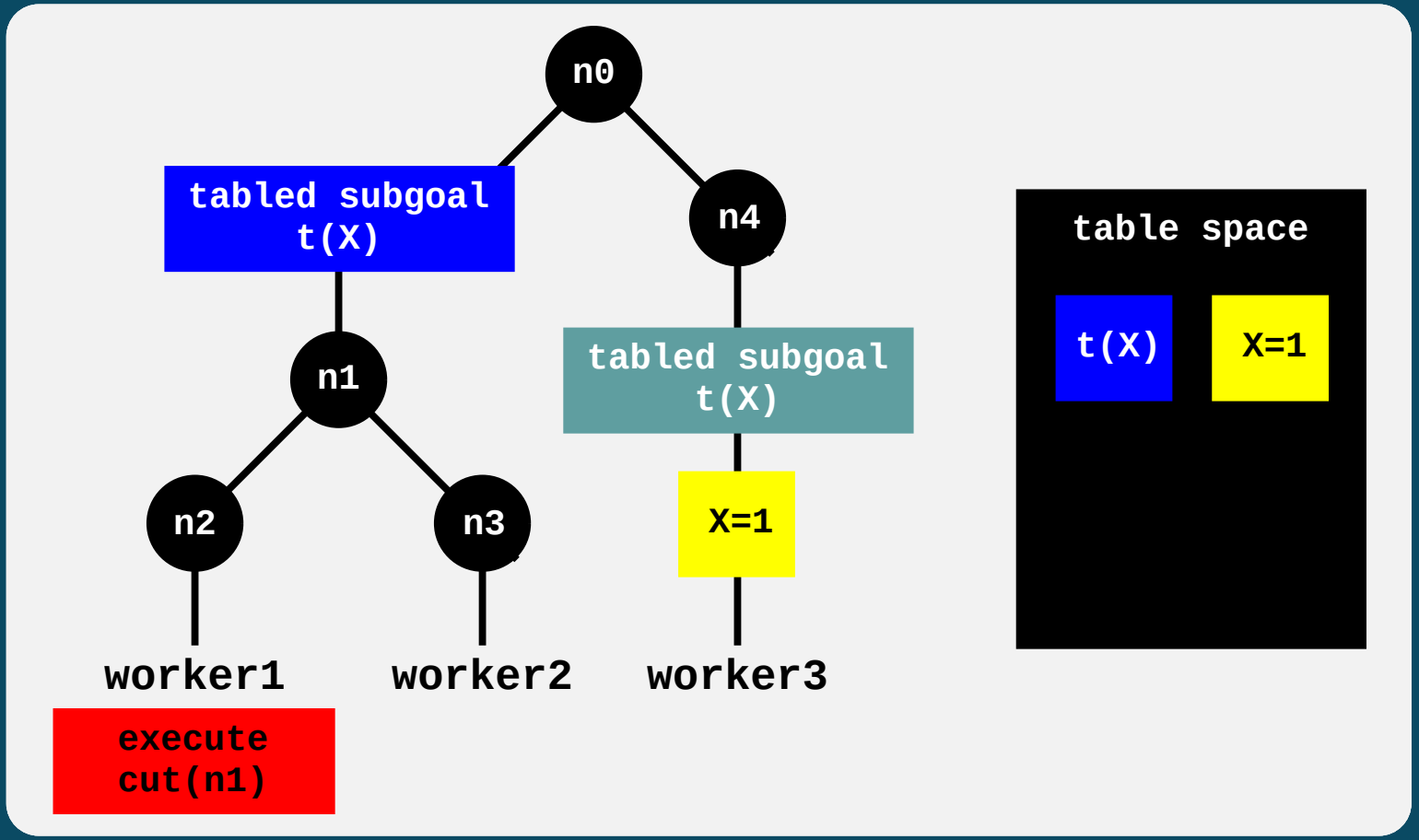
Tabling and Speculative Computations

- **Problem II** - Speculative answers may generate further speculative computations.



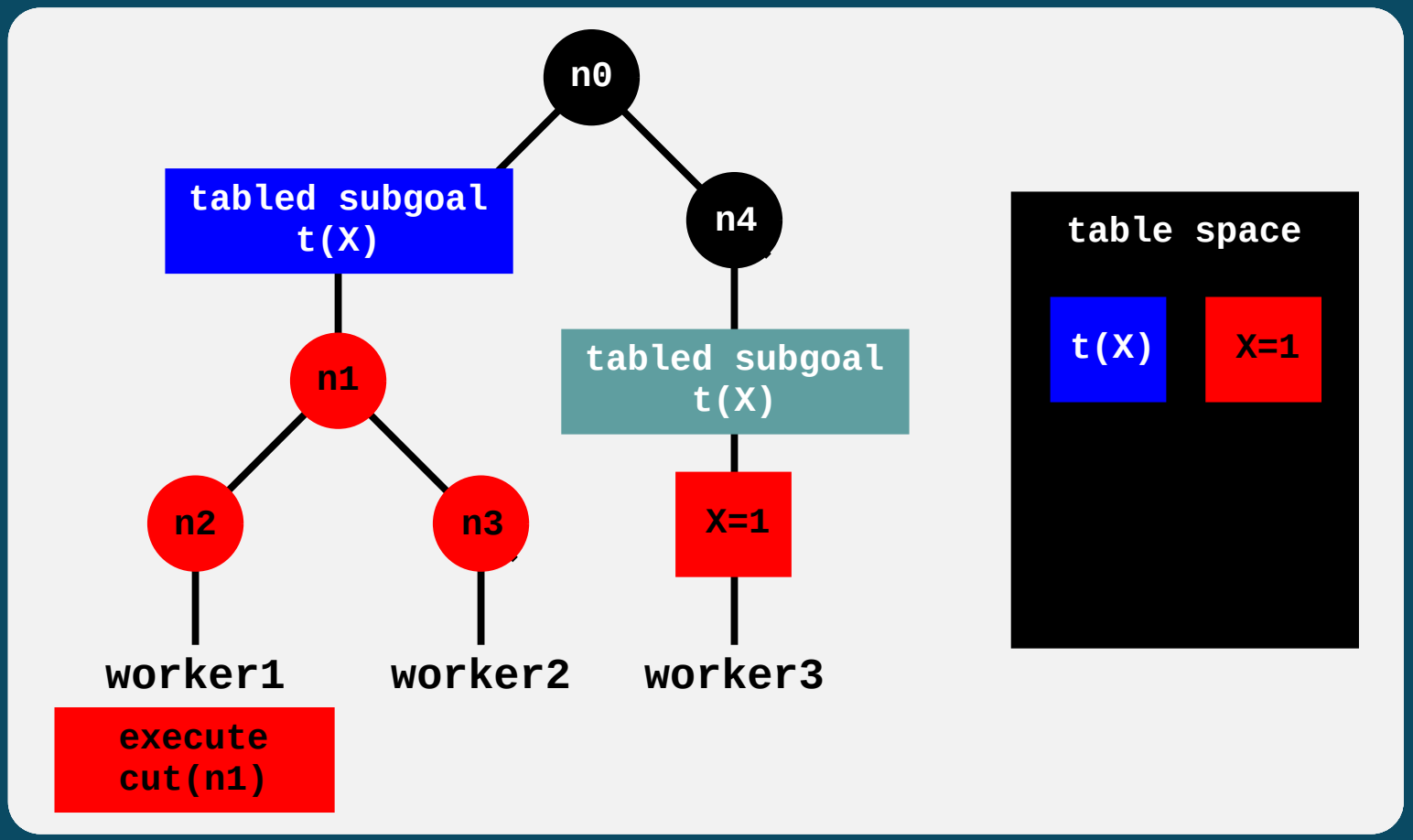
Tabling and Speculative Computations

- **Problem II** - Speculative answers may generate further speculative computations.



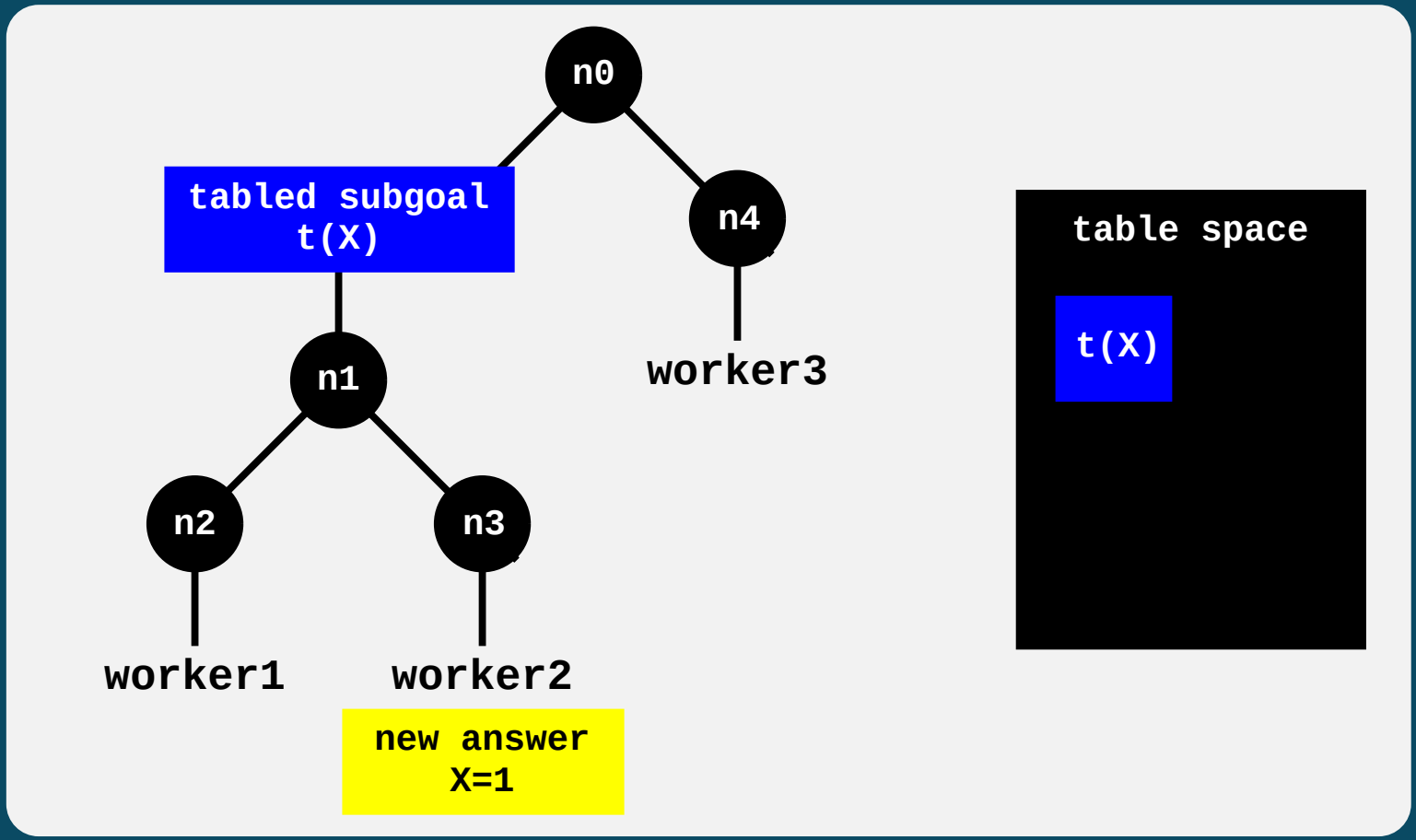
Tabling and Speculative Computations

- **Problem II** - Speculative answers may generate further speculative computations.



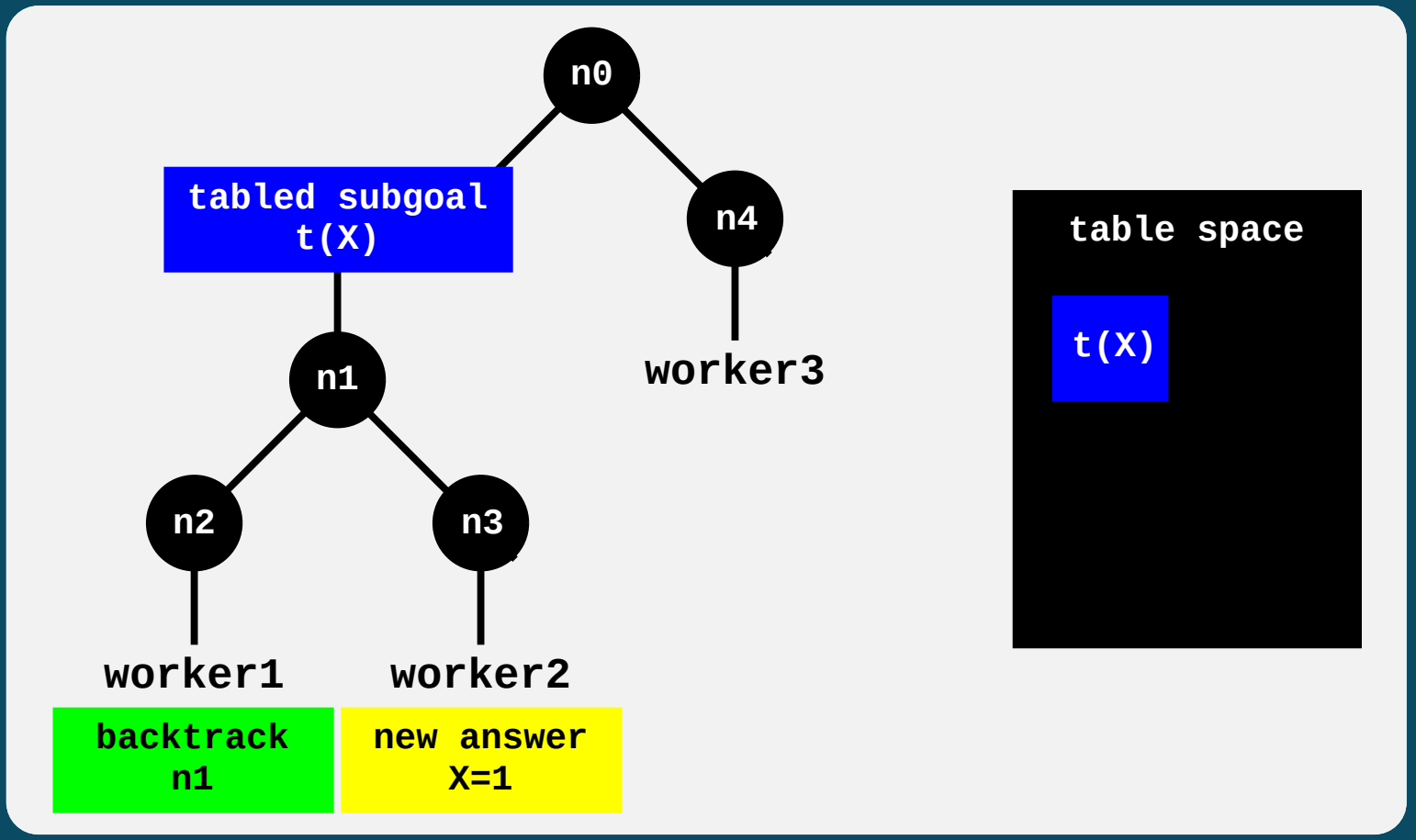
Tabling and Speculative Computations

➤ **Problem III** - Delaying the availability of answers may compromise performance.



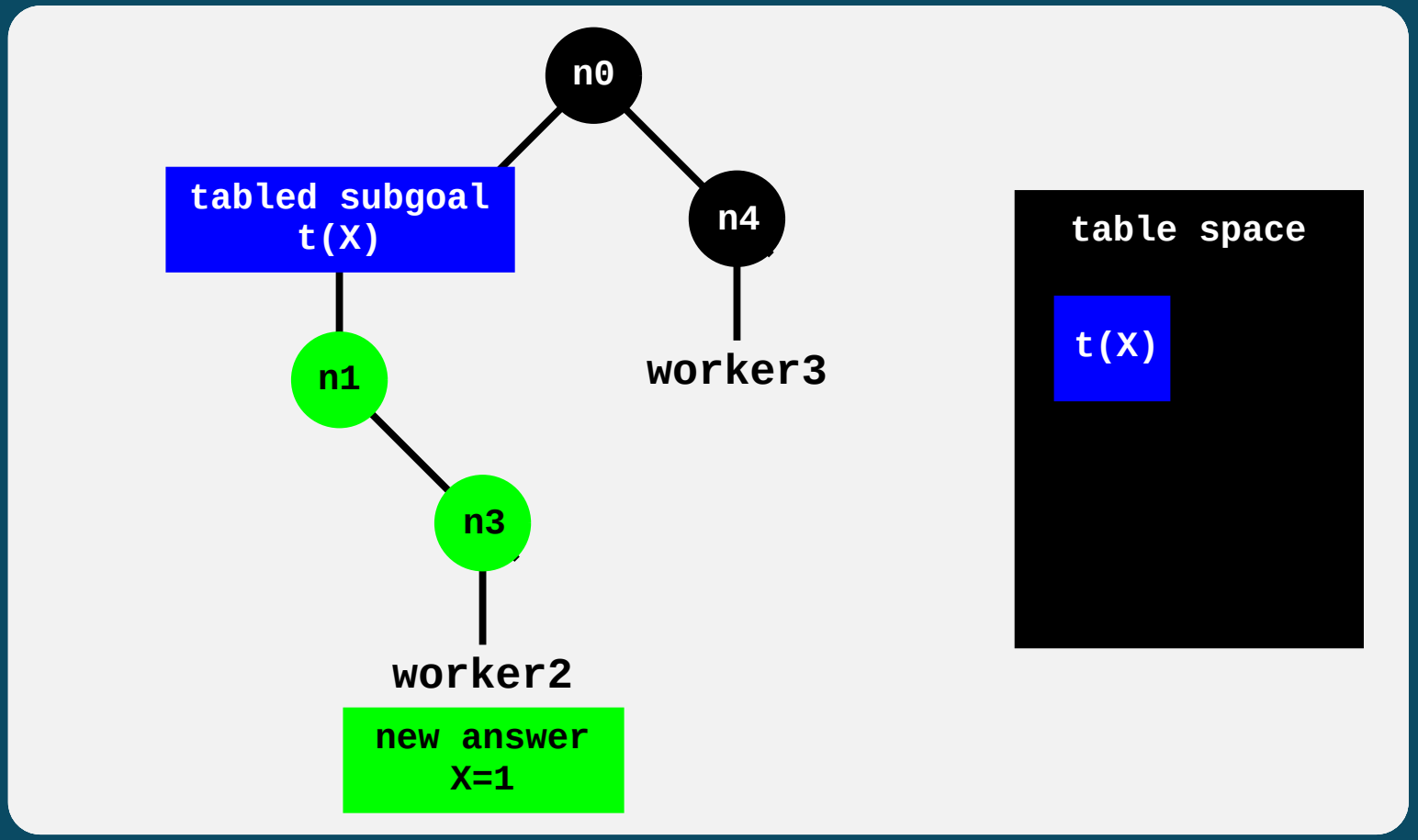
Tabling and Speculative Computations

➤ **Problem III** - Delaying the availability of answers may compromise performance.



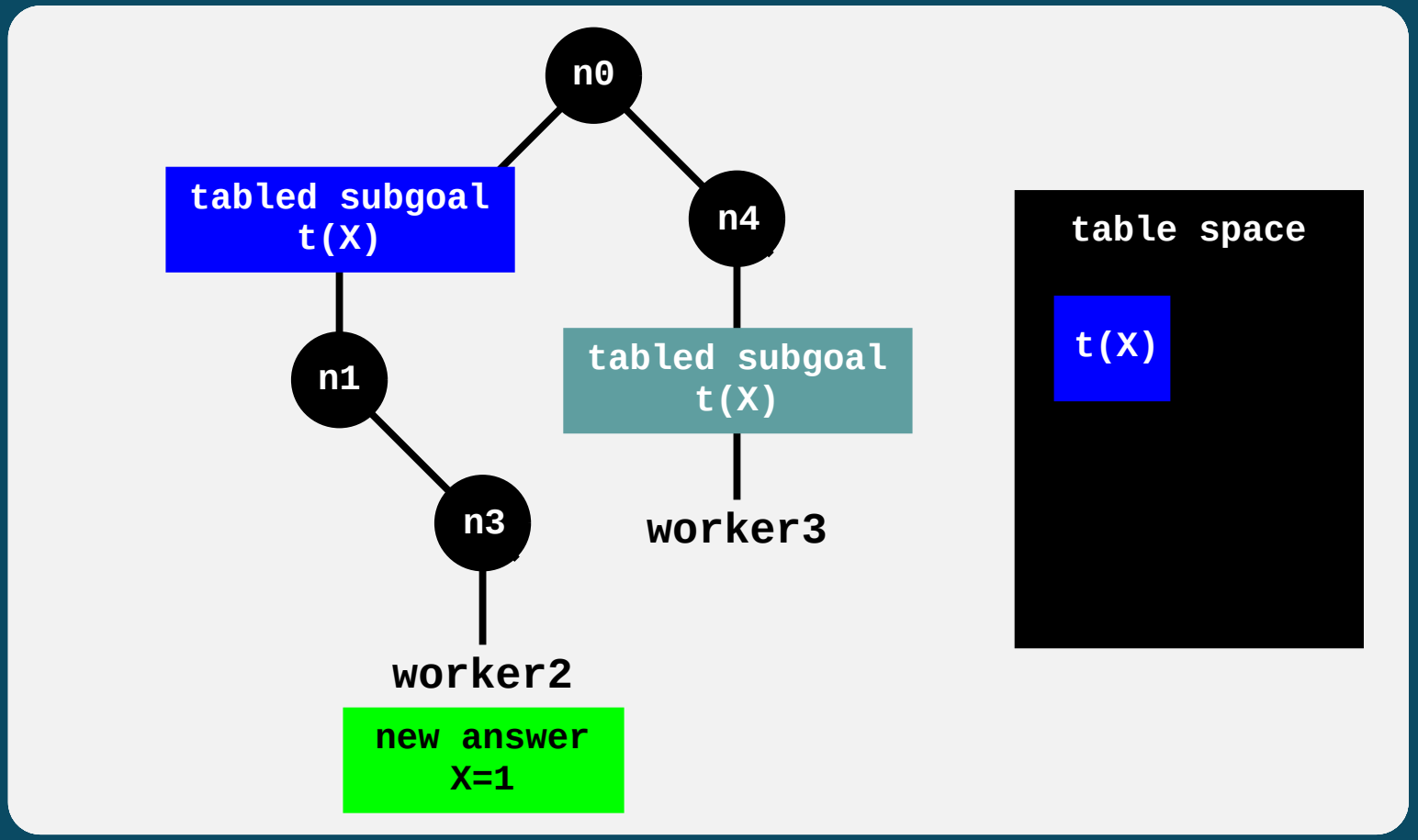
Tabling and Speculative Computations

➤ **Problem III** - Delaying the availability of answers may compromise performance.



Tabling and Speculative Computations

- **Problem III** - Delaying the availability of answers may compromise performance.



Our Approach

- Guiding rules:
  - **Rule 1** - Answers cannot be tabled until they are safe from being pruned.
  - **Rule 2** - Answers should be made available as soon as it is proved that they are not speculative.
  - **Rule 3** - Avoid explicit communication/synchronization between workers.

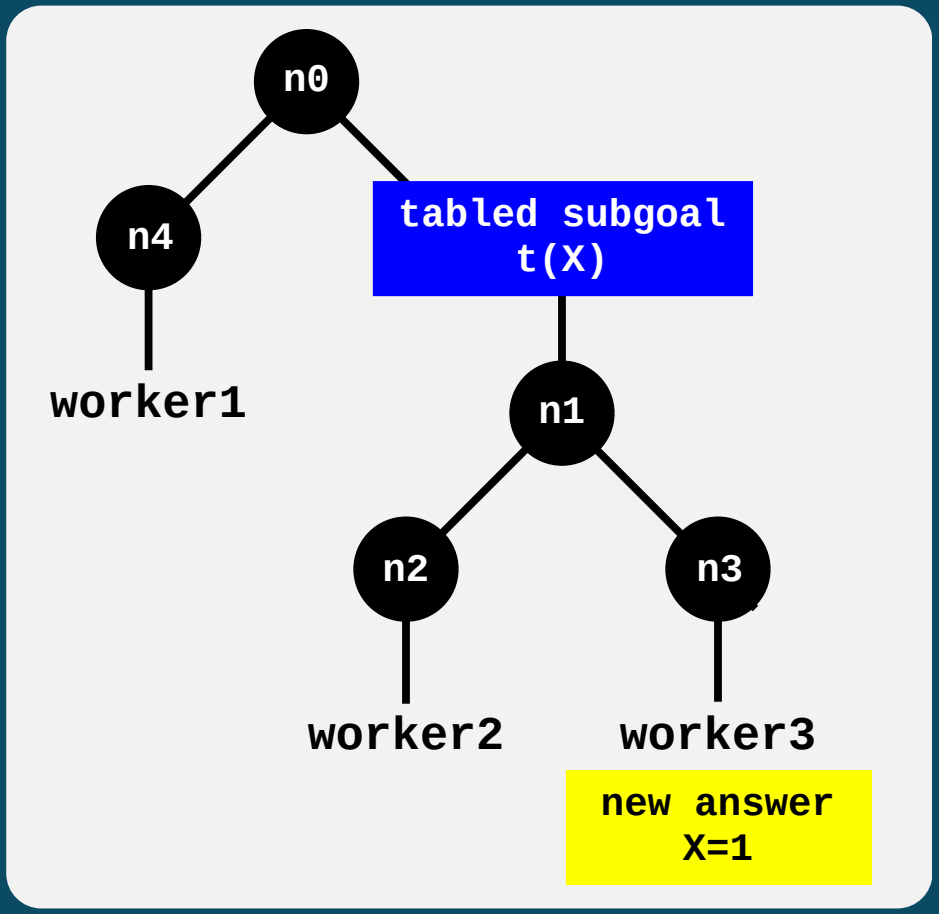
Our Approach

- Guiding rules:
  - **Rule 1** - Answers cannot be tabled until they are safe from being pruned.
  - **Rule 2** - Answers should be made available as soon as it is proved that they are not speculative.
  - **Rule 3** - Avoid explicit communication/synchronization between workers.

- Key idea:
  - Allow speculative answers to be stored in advance into the table.
  - Delay their visibility by leaving marks in the youngest nodes that can potentially prune the branches where they were found.

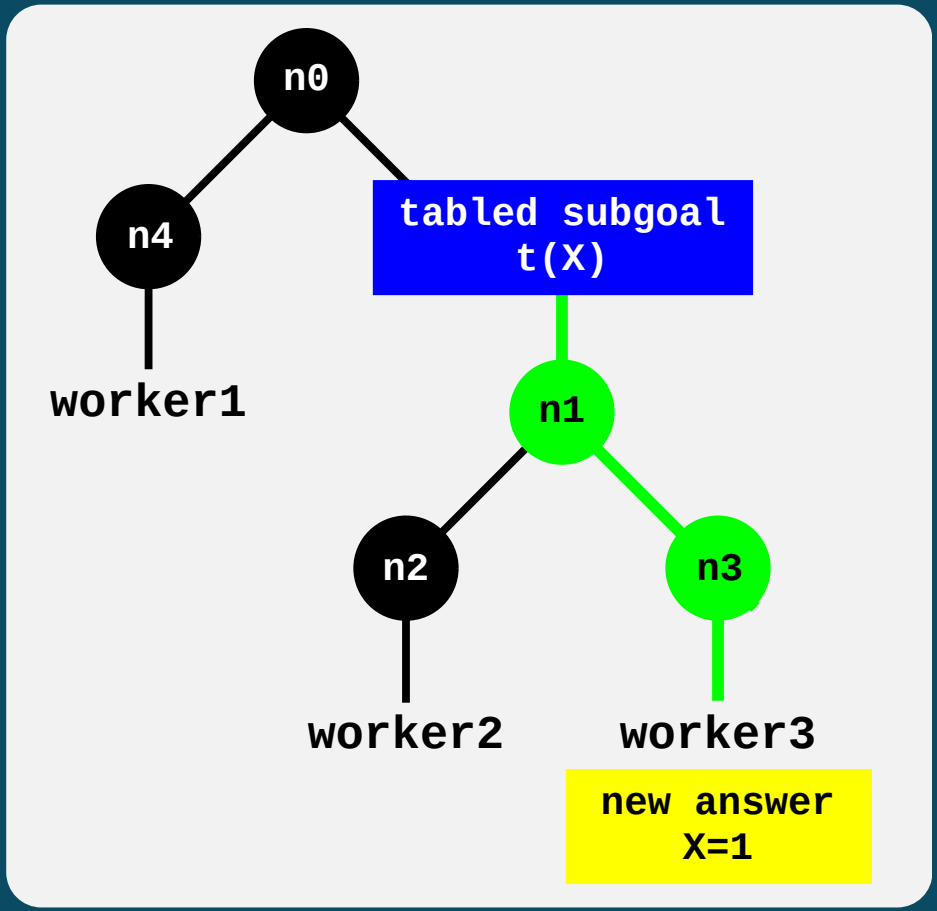
Detecting Speculative Tabled Answers

- To detect if a tabled answer is speculative we need to check the safeness of the branch where it was found.



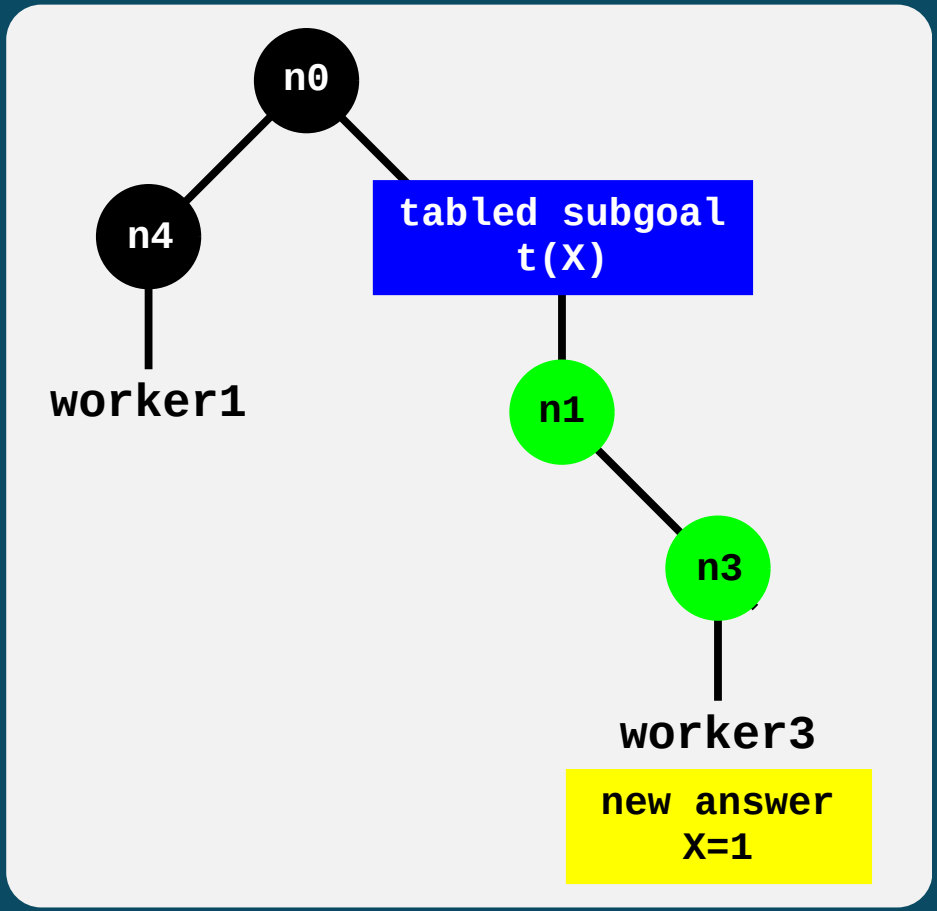
Detecting Speculative Tabled Answers

- To detect if a tabled answer is speculative we need to check the safeness of the branch where it was found.
- The branch only needs to be checked until the generator node, as otherwise it would be an outer cut operation.



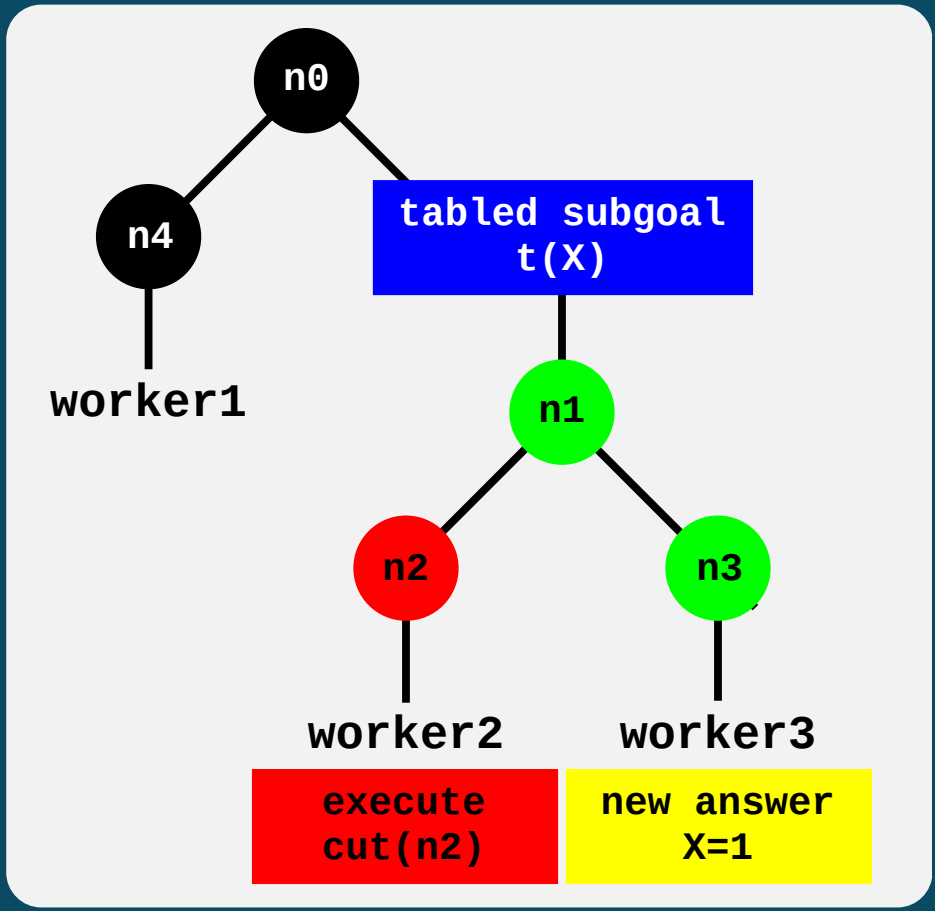
Detecting Speculative Tabled Answers

- To detect if a tabled answer is speculative we need to check the safeness of the branch where it was found.
- The branch only needs to be checked until the generator node, as otherwise it would be an outer cut operation.
- A branch is safe if:
  - It is leftmost;



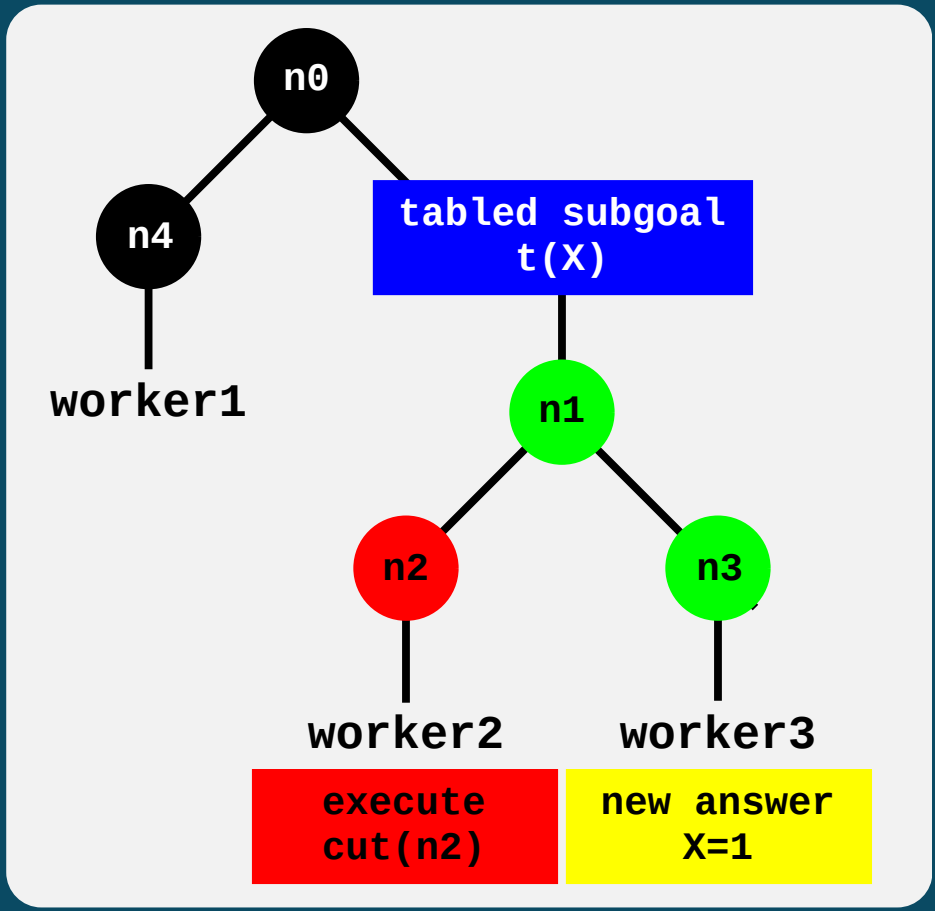
Detecting Speculative Tabled Answers

- To detect if a tabled answer is speculative we need to check the safeness of the branch where it was found.
- The branch only needs to be checked until the generator node, as otherwise it would be an outer cut operation.
- A branch is safe if:
  - It is leftmost;
  - Workers in branches to the left cannot prune it.



Detecting Speculative Tabled Answers

- To detect if a tabled answer is speculative we need to check the safeness of the branch where it was found.
- The branch only needs to be checked until the generator node, as otherwise it would be an outer cut operation.
- A branch is safe if:
  - It is leftmost;
  - Workers in branches to the left cannot prune it.



- OPTYap already includes most of the machinery to perform such detection.
- However, we still need a mechanism to calculate the nodes that can potentially be pruned by a worker.

Detecting Speculative Tabled Answers

- We added the following extensions to OPTYap:
  - A **GLOBAL_pruning_workers** bitmap to keep track of the workers that are executing branches with cuts.
  - A **LOCAL_safe_scope** register for each worker to reference the youngest node that cannot be pruned away by the worker.
  - A **clause_with_cuts** instruction to mark the clauses with cuts. It is included during compilation as the first instruction to be executed for such clauses.

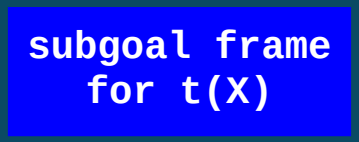
Detecting Speculative Tabled Answers

- We added the following extensions to OPTYap:
  - A **GLOBAL_pruning_workers** bitmap to keep track of the workers that are executing branches with cuts.
  - A **LOCAL_safe_scope** register for each worker to reference the youngest node that cannot be pruned away by the worker.
  - A **clause_with_cuts** instruction to mark the clauses with cuts. It is included during compilation as the first instruction to be executed for such clauses.

```
clause_with_cuts() {
   if (LOCAL_safe_scope == NULL) {
      insert_into_bitmap(GLOBAL_pruning_workers, WORKER_id)
      LOCAL_safe_scope = B            // B points to the current choice point
   } else if (LOCAL_safe_scope is younger than B) {
      LOCAL_safe_scope = B
   }
}
```

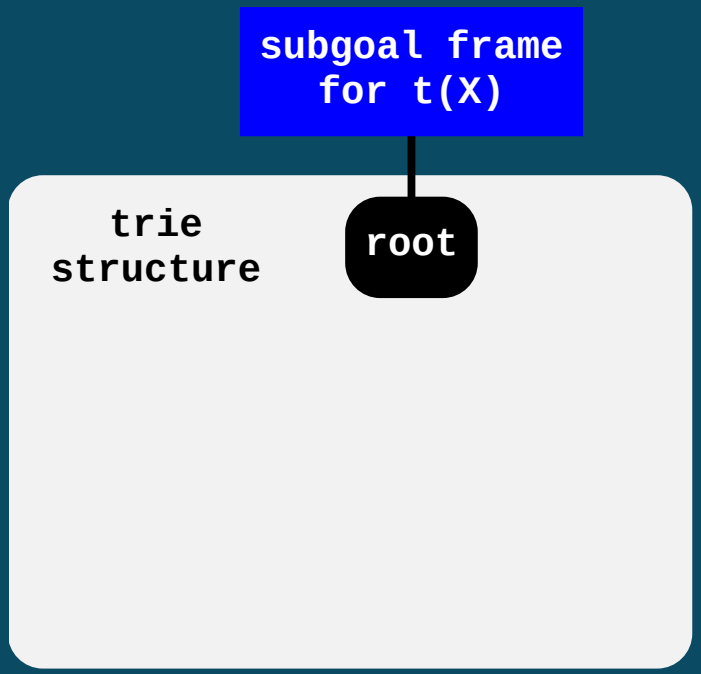
Storing Answers in the Table Space

- A subgoal frame is the entry point for the subgoal answers.



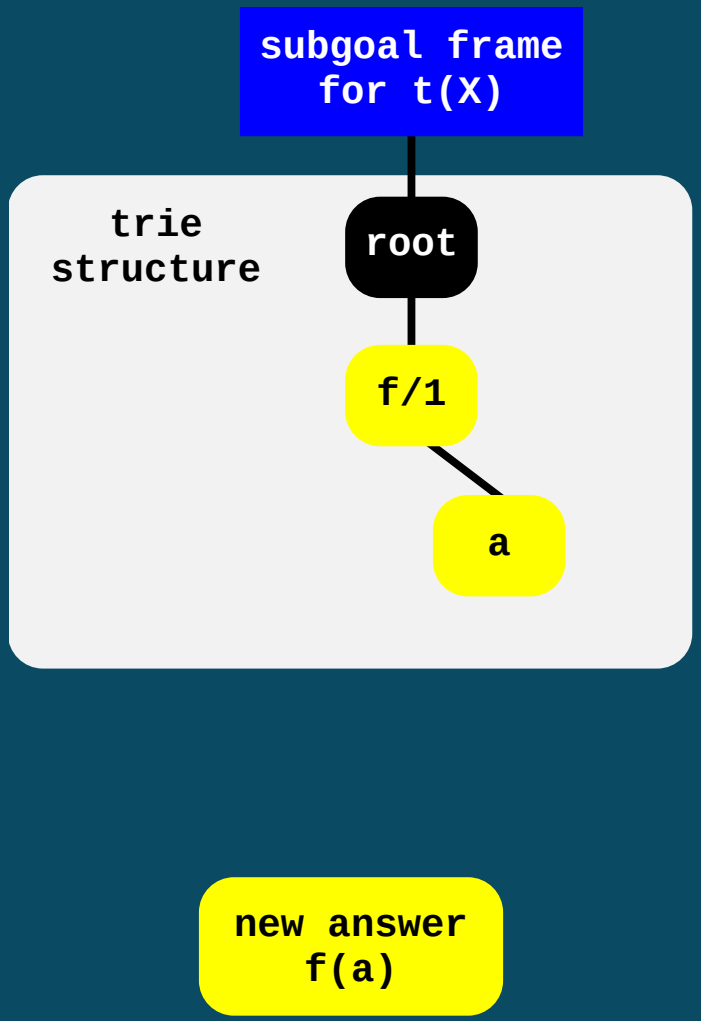
Storing Answers in the Table Space

- A subgoal frame is the entry point for the subgoal answers.
- Answers are represented by tries. Tries are trees in which common prefixes are represented only once.



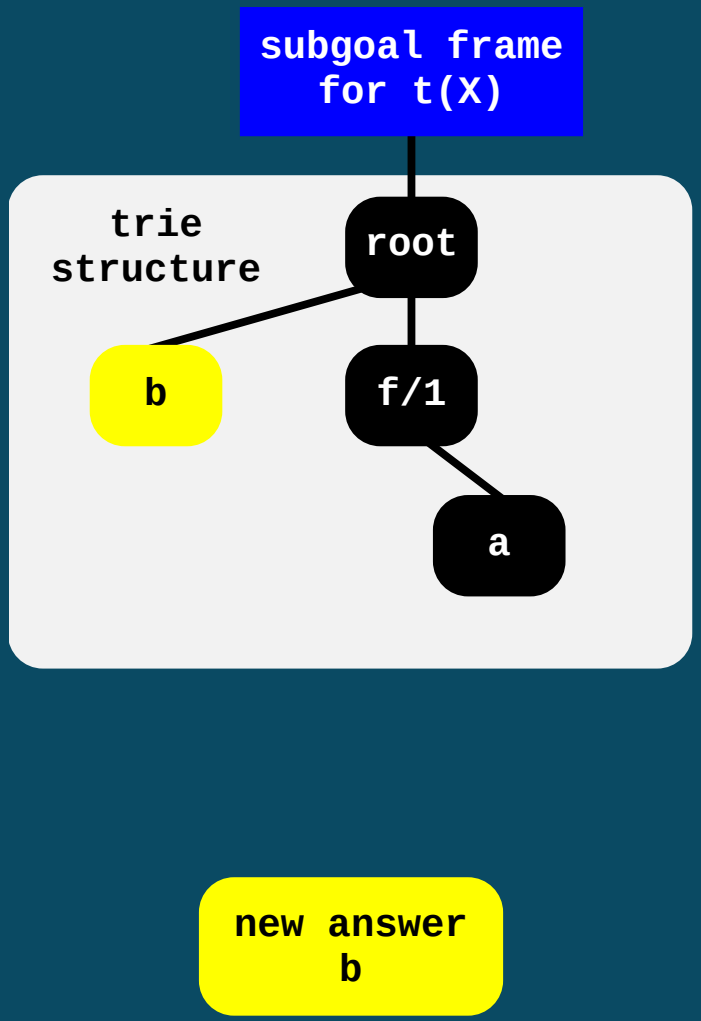
Storing Answers in the Table Space

- A subgoal frame is the entry point for the subgoal answers.
- Answers are represented by tries. Tries are trees in which common prefixes are represented only once.
- Each different path through the nodes in the trie corresponds to an answer.



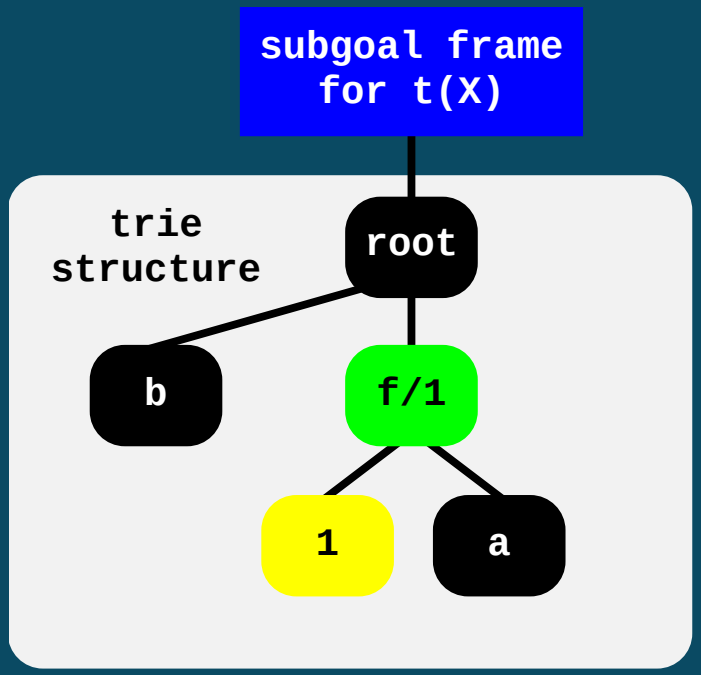
Storing Answers in the Table Space

- A subgoal frame is the entry point for the subgoal answers.
- Answers are represented by tries. Tries are trees in which common prefixes are represented only once.
- Each different path through the nodes in the trie corresponds to an answer.



Storing Answers in the Table Space

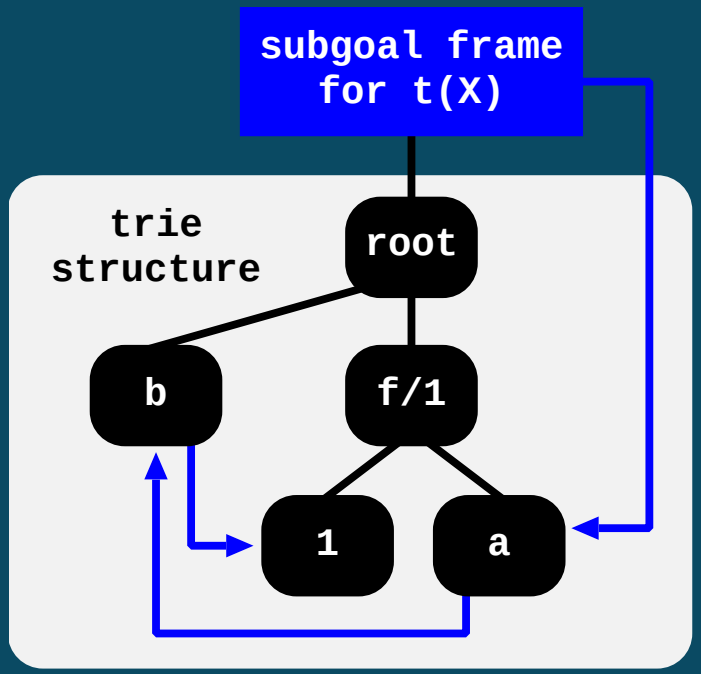
- A subgoal frame is the entry point for the subgoal answers.
- Answers are represented by tries. Tries are trees in which common prefixes are represented only once.
- Each different path through the nodes in the trie corresponds to an answer.
- Terms with common prefixes branch off from each other at the first distinguishing symbol.



new answer
f(1)

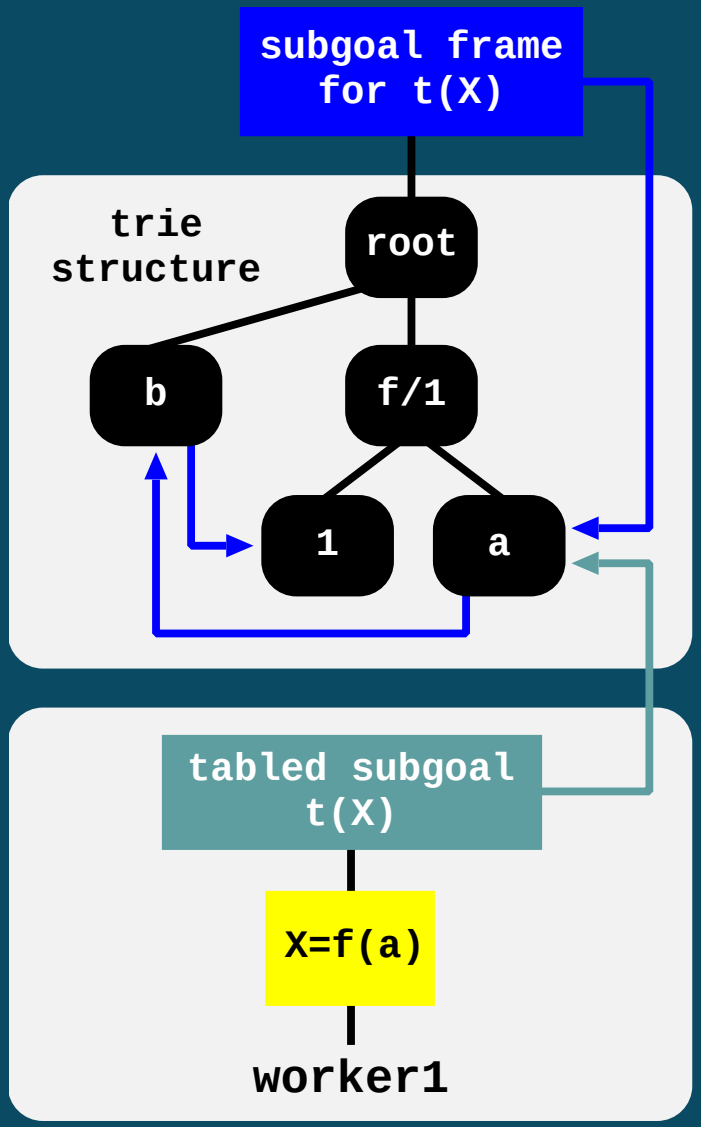
Storing Answers in the Table Space

- A subgoal frame is the entry point for the subgoal answers.
- Answers are represented by tries. Tries are trees in which common prefixes are represented only once.
- Each different path through the nodes in the trie corresponds to an answer.
- Terms with common prefixes branch off from each other at the first distinguishing symbol.
- Leaf nodes are chained in insertion time order.



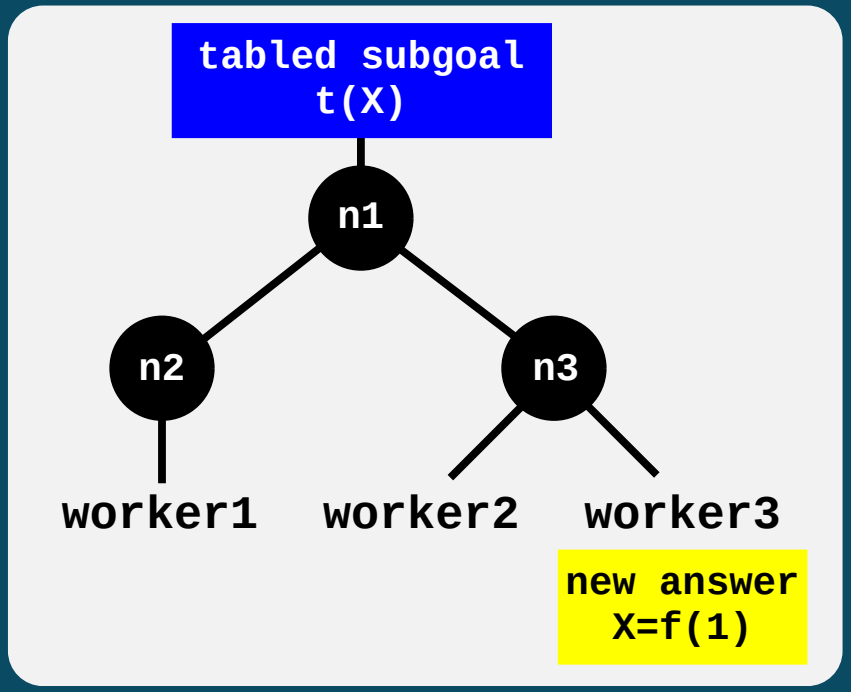
Storing Answers in the Table Space

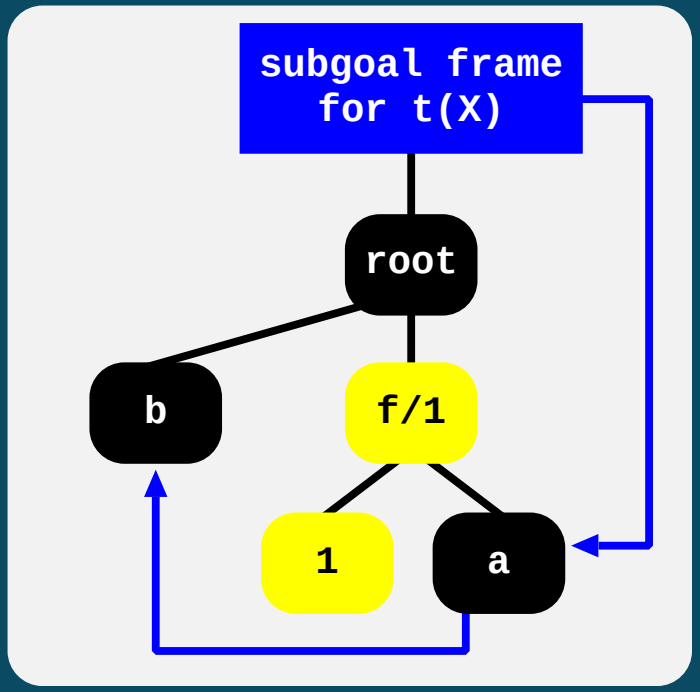
- A subgoal frame is the entry point for the subgoal answers.
- Answers are represented by tries. Tries are trees in which common prefixes are represented only once.
- Each different path through the nodes in the trie corresponds to an answer.
- Terms with common prefixes branch off from each other at the first distinguishing symbol.
- Leaf nodes are chained in insertion time order.Variant calls keep a reference to the leaf node of the last consumed answer, hence can consume more answers by following the chain.



Pending Tabled Answers

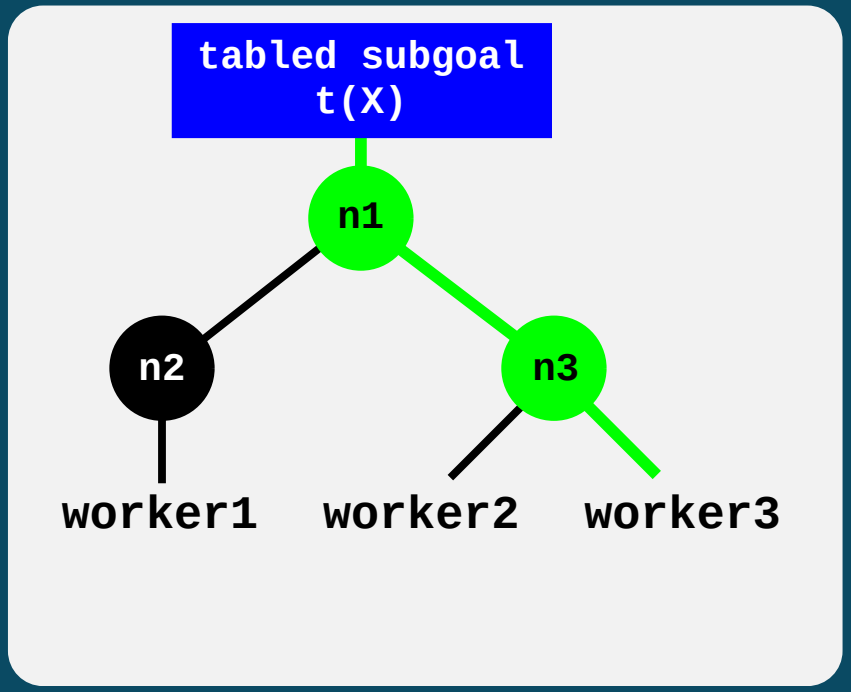
- When a new answer is found, we insert it in advance into the table space.

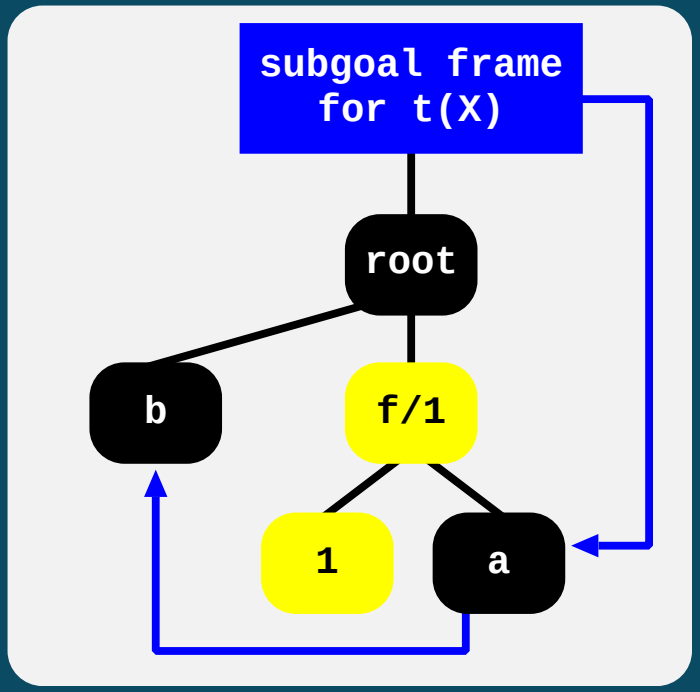




Pending Tabled Answers

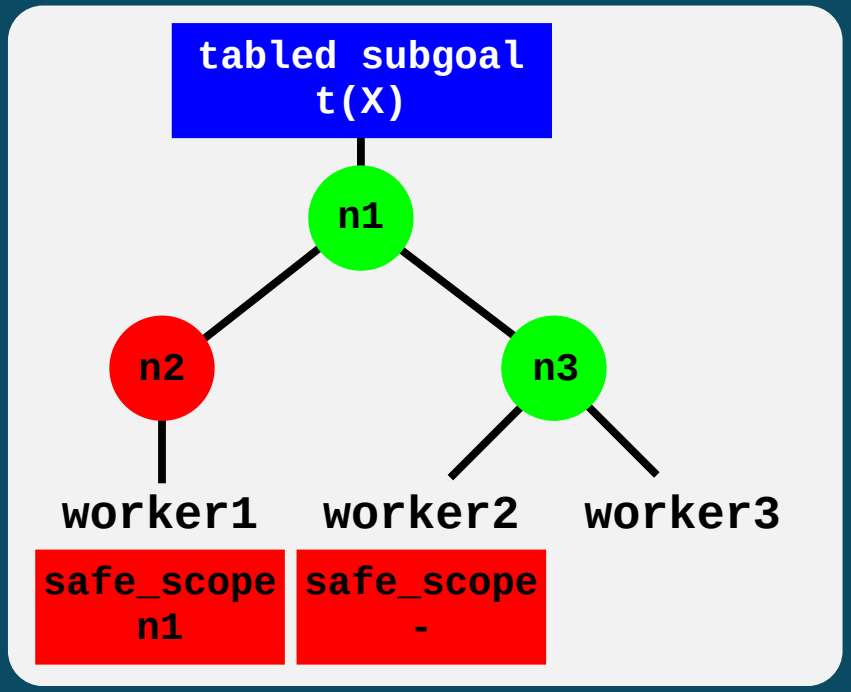
- When a new answer is found, we insert it in advance into the table space.
- Then, we check the safeness of the branch where the answer was found:

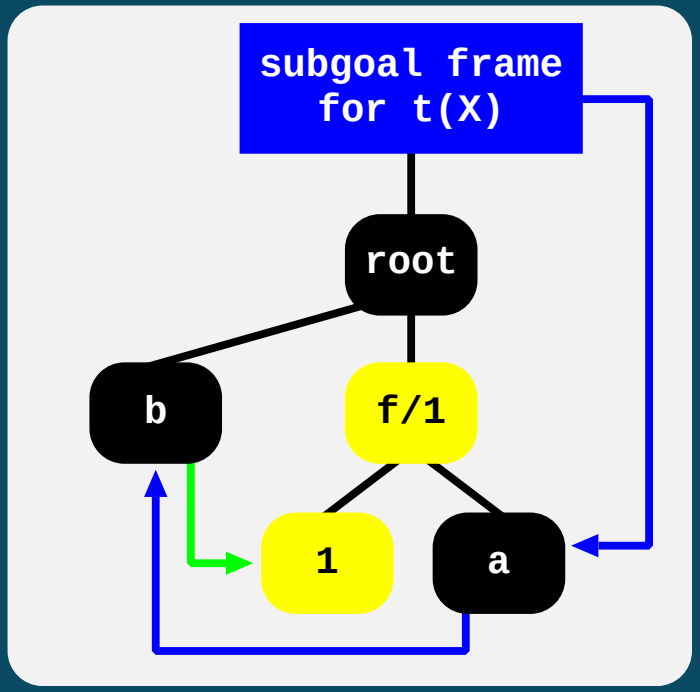




Pending Tabled Answers

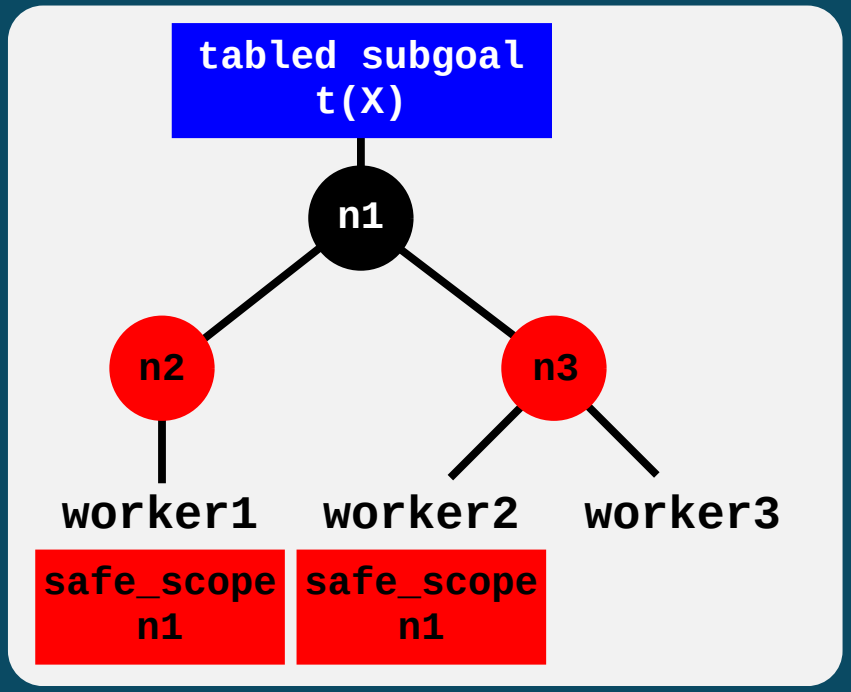
- When a new answer is found, we insert it in advance into the table space.
- Then, we check the safeness of the branch where the answer was found:
  - Being safe, the answer is added to the chain of available answers, as usual.

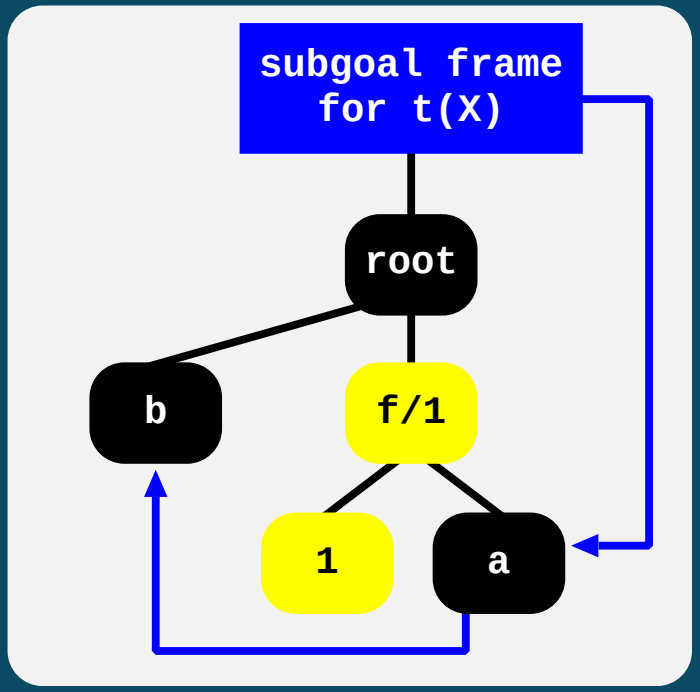




Pending Tabled Answers

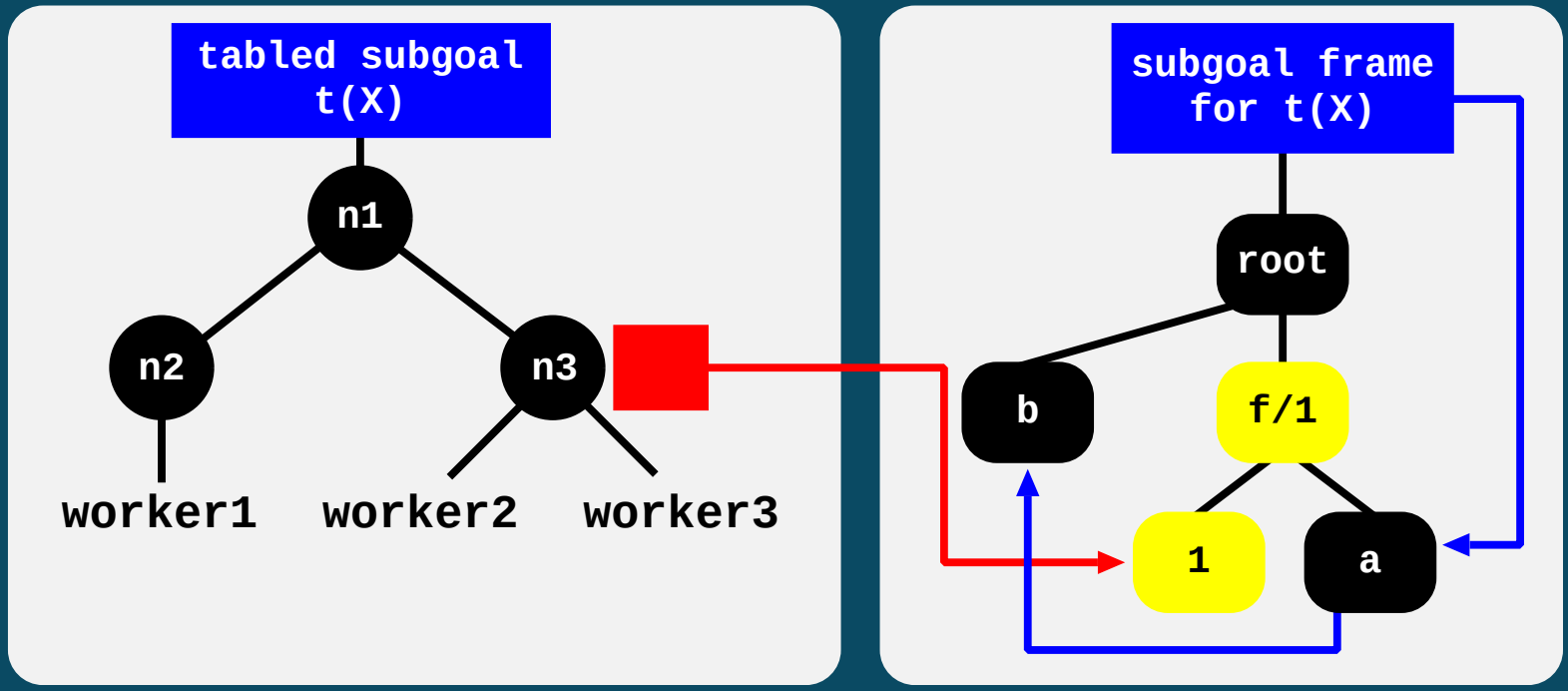
- When a new answer is found, we insert it in advance into the table space.
- Then, we check the safeness of the branch where the answer was found:
  - Being safe, the answer is added to the chain of available answers, as usual.
  - Being speculative





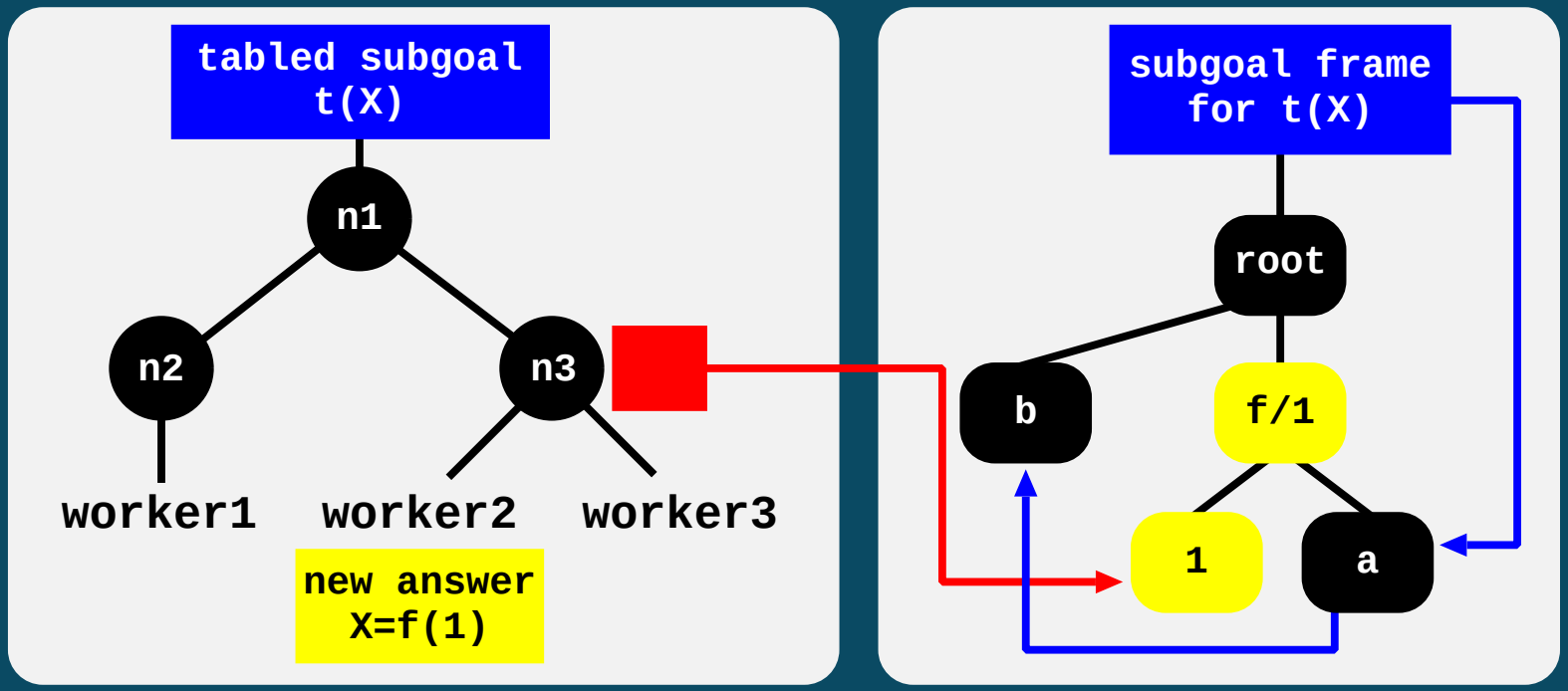
Pending Tabled Answers

- When a new answer is found, we insert it in advance into the table space.
- Then, we check the safeness of the branch where the answer was found:
  - Being safe, the answer is added to the chain of available answers, as usual.
  - Being speculative, it is left pending in the youngest node where it can be pruned away by a worker in a left branch.



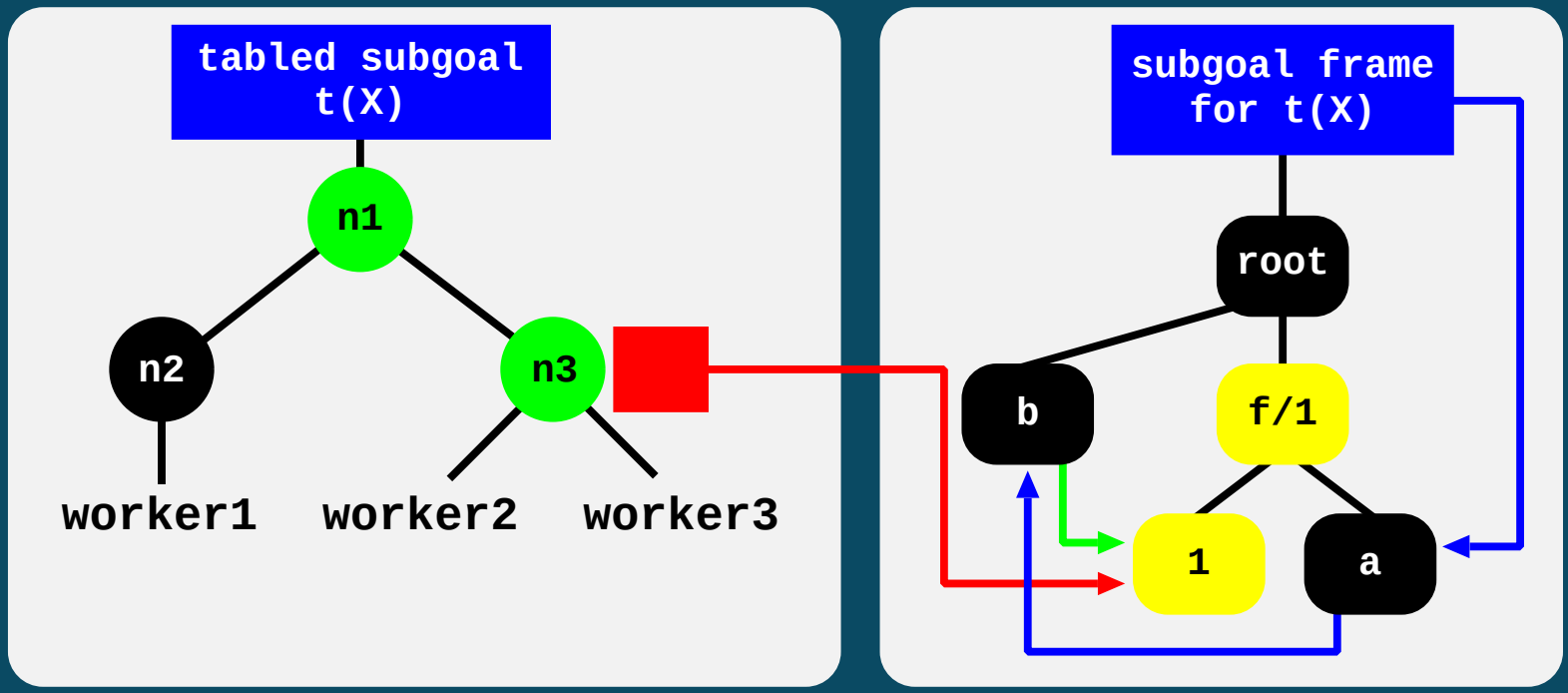
Pending Tabled Answers

- Consider now that a new occurrence of `f(1)` is found elsewhere.



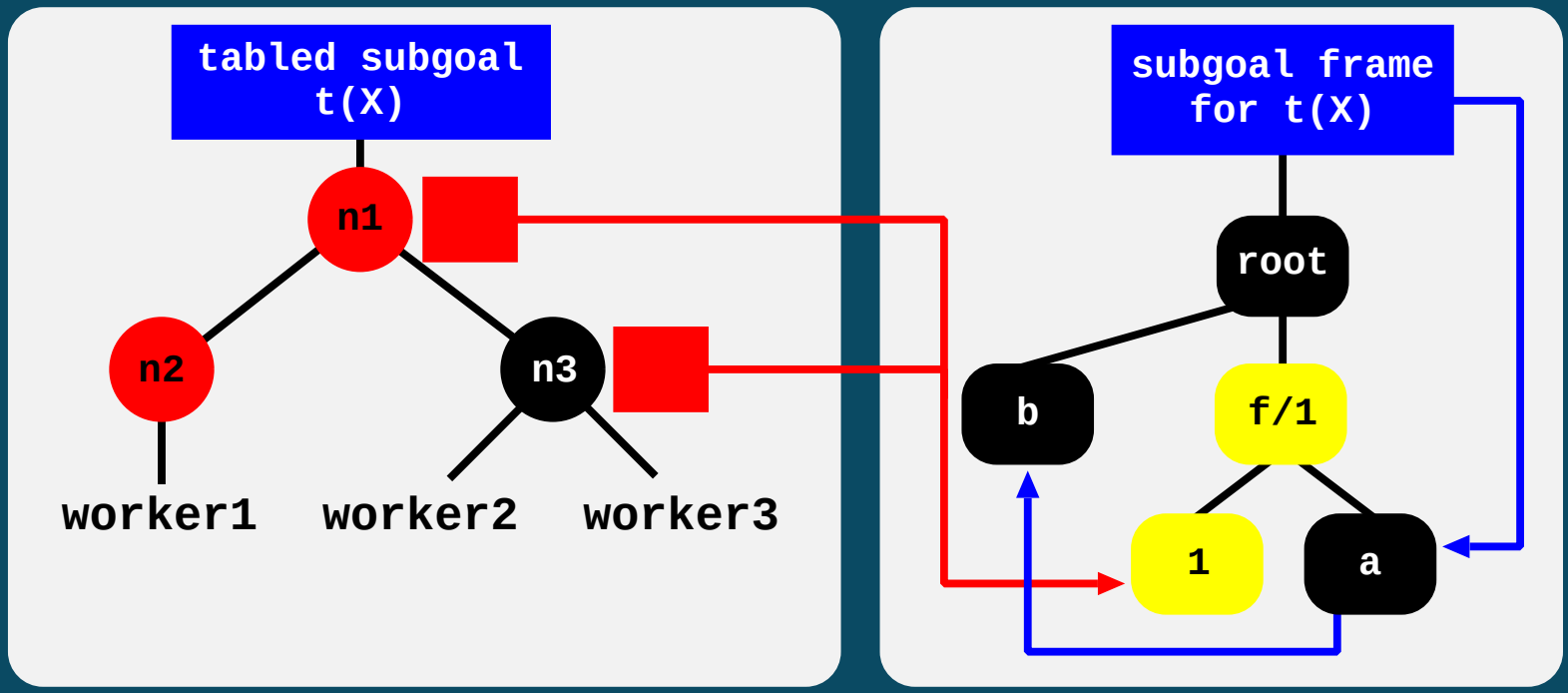
Pending Tabled Answers

- Consider now that a new occurrence of `f(1)` is found elsewhere.
- Again two situations may occur:
  - If the branch is safe, the answer is added to the chain of available answers.



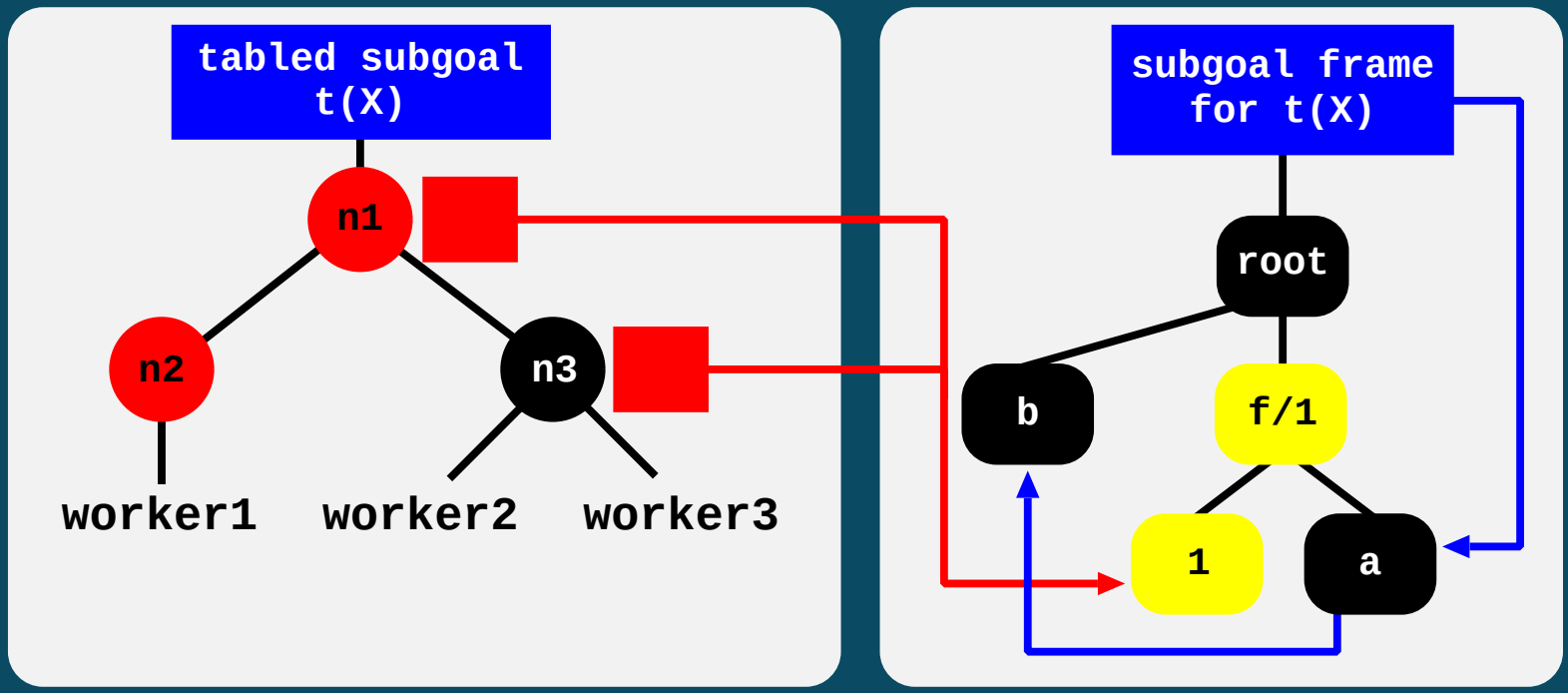
Pending Tabled Answers

- Consider now that a new occurrence of `f(1)` is found elsewhere.
- Again two situations may occur:
  - If the branch is safe, the answer is added to the chain of available answers.
  - If speculative, the answer is left pending. This is necessary because there is no way to know beforehand in which branch it will be safe first, if in any.



Pending Tabled Answers

- Consider now that a new occurrence of `f(1)` is found elsewhere.
- Again two situations may occur:
  - If the branch is safe, the answer is added to the chain of available answers.
  - If speculative, the answer is left pending. This is necessary because there is no way to know beforehand in which branch it will be safe first, if in any.



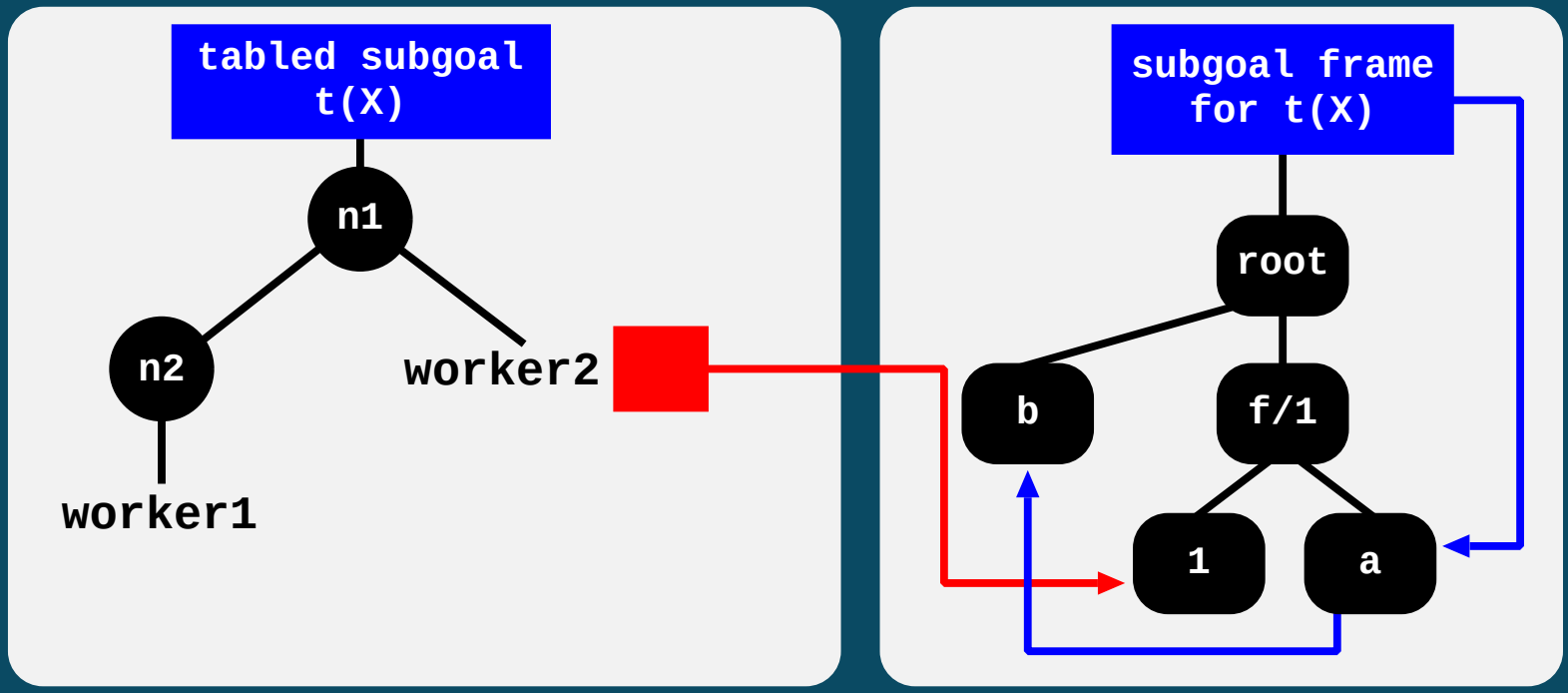
- Note that in both cases, `f(1)` is only represented once in the table space.

Pruning Pending Tabled Answers

- When a node $\mathcal{N}$ is pruned, its pending tabled answers can be:
  - Only left pending in $\mathcal{N}$;
  - Also left pending in other nodes;
  - Already valid answers.
- Note that for all three situations no interaction with other workers or with the table space is needed and $\mathcal{N}$ can simply be pruned away.
- Even for the first situation, we may keep the answers in the table as in the meantime they can still be found in other branches.

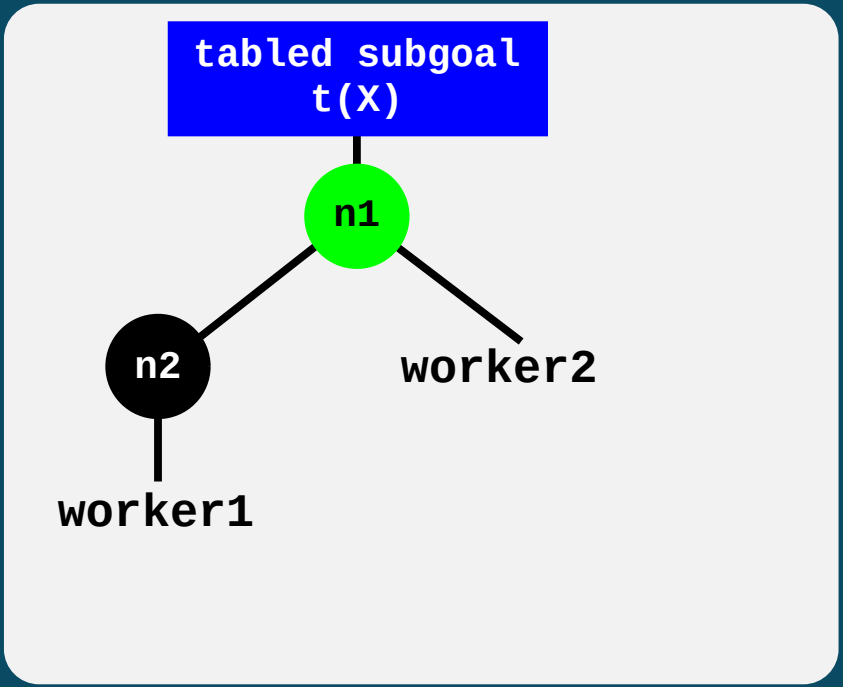
Releasing Pending Tabled Answers

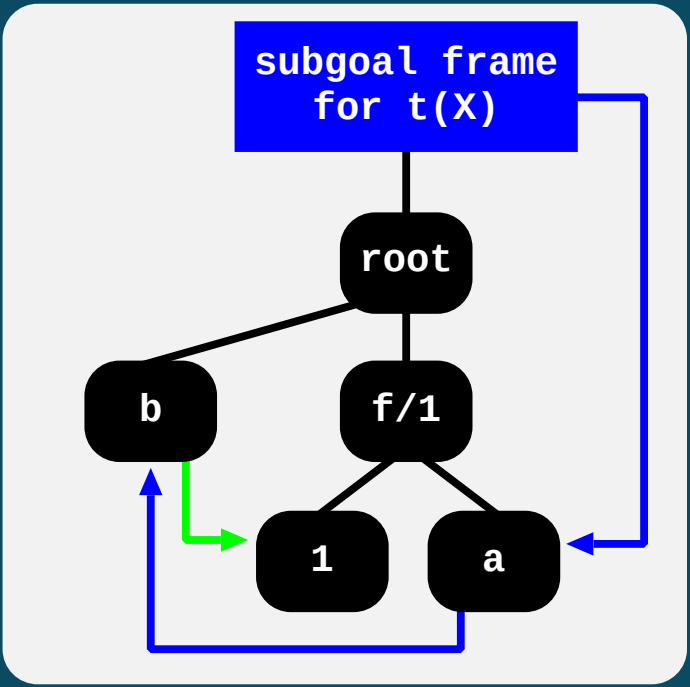
- The last worker leaving a node with pending tabled answers, determines the next youngest node $\mathcal{N}$ where it can be pruned by a worker in a left branch.



Releasing Pending Tabled Answers

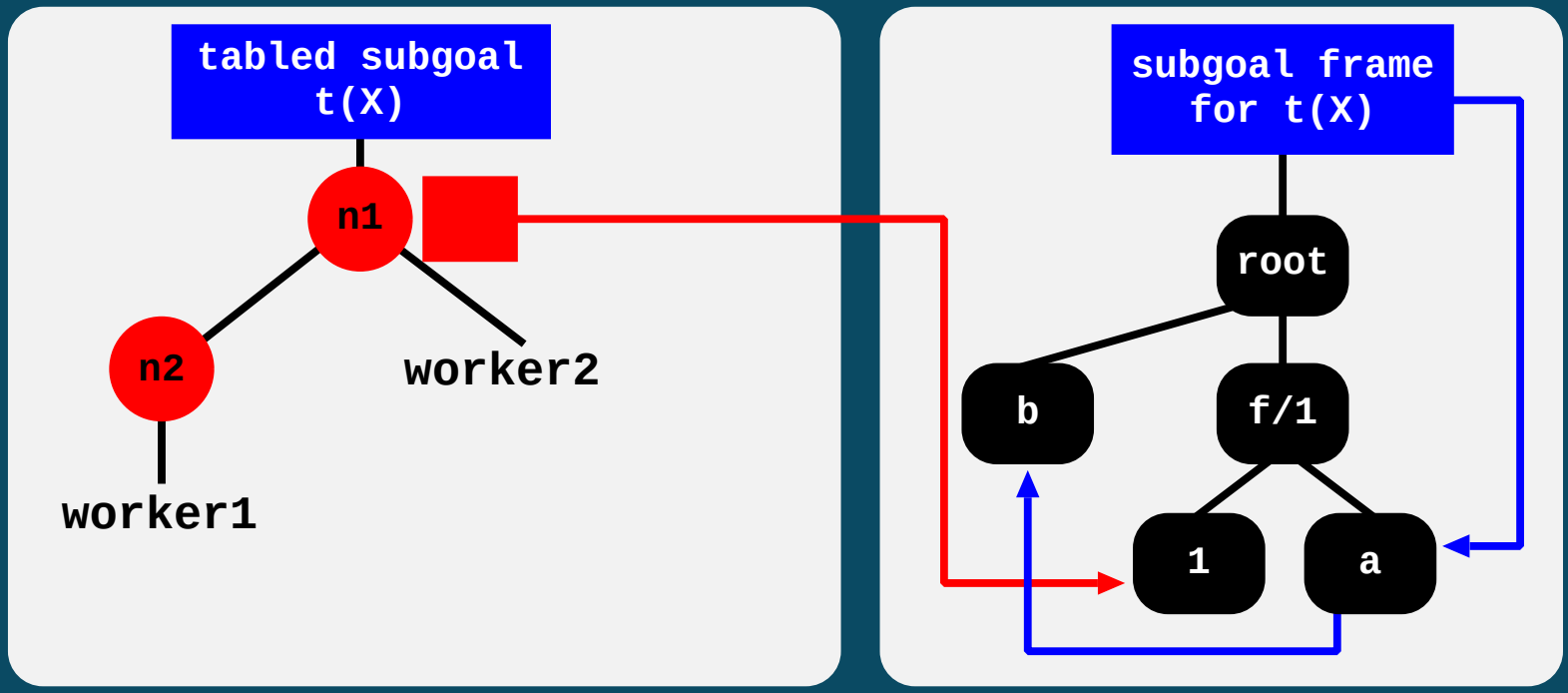
- The last worker leaving a node with pending tabled answers, determines the next youngest node $\mathcal{N}$ where it can be pruned by a worker in a left branch.
  - Pending answers corresponding to generator nodes younger than $\mathcal{N}$ are made visible.





Releasing Pending Tabled Answers

- The last worker leaving a node with pending tabled answers, determines the next youngest node $\mathcal{N}$ where it can be pruned by a worker in a left branch.
  - Pending answers corresponding to generator nodes younger than $\mathcal{N}$ are made visible.
  - Otherwise, are left pending in $\mathcal{N}$.



Conclusions

- We discussed the problem of speculative computations in or-parallel tabling.
- Our approach deals with speculative tabled computations by delaying the point at which answers are made visible in the table.
- Speculative answers are stored in advance into the table space and left pending in the youngest nodes that can potentially prune the branches where they were found. No explicit communication/synchronization between workers is required.
- We have designed simple extensions to:
  - Calculate the nodes that can be potentially pruned by each worker.
  - Keep track of left pending answers;
- With this support, OPTYap is now able to execute a wider range of applications without introducing significant overheads (**less than 1%**) for applications without cuts.