A Tabling Engine Designed to Support Mixed-Strategy Evaluation

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

Motivation

Motivation

- During tabled execution, there are several points where we can choose between continuing forward execution, backtracking, returning answers to consumer nodes, or performing completion.
- The decision on which operation to perform depends on the **scheduling strategy**.

Motivation

➤ Motivation

- ♦ During tabled execution, there are several points where we can choose between continuing forward execution, backtracking, returning answers to consumer nodes, or performing completion.
- ♦ The decision on which operation to perform depends on the **scheduling strategy**.
- ♦ A strategy can achieve very good performance for certain applications, but for others it might add overheads and lead to inefficiency [Freire 97].
- ♦ Achieving the best performance can be done if we have the ability to use **multiple strategies** within the same evaluation [Freire, Warren 1997].

Motivation

➤ Motivation

- ♦ During tabled execution, there are several points where we can choose between continuing forward execution, backtracking, returning answers to consumer nodes, or performing completion.
- ♦ The decision on which operation to perform depends on the **scheduling strategy**.
- ♦ A strategy can achieve very good performance for certain applications, but for others it might add overheads and lead to inefficiency [Freire 97].
- ♦ Achieving the best performance can be done if we have the ability to use **multiple strategies** within the same evaluation [Freire, Warren 1997].

➤ Our Goal

- ♦ Design a tabling engine that supports simultaneous **mixed-strategy evaluation** for the two most successful tabling scheduling strategies: **batched and local scheduling**.

Basic Tabling Definitions

Basic Execution Model

- Whenever a tabled subgoal is first called, a new entry is allocated in the **table space**. This entry will collect all the answers generated for the subgoal.
- Variant calls to tabled subgoals are resolved by consuming the answers already stored in the table.
- Meanwhile, as new answers are found, they are inserted into the table and returned to all variant subgoals.

Basic Tabling Definitions

➤ Basic Execution Model

- ♦ Whenever a tabled subgoal is first called, a new entry is allocated in the **table space**. This entry will collect all the answers generated for the subgoal.
- ♦ Variant calls to tabled subgoals are resolved by consuming the answers already stored in the table.
- ♦ Meanwhile, as new answers are found, they are inserted into the table and returned to all variant subgoals.

➤ Nodes Classification

- ♦ **Generator**: nodes that first call a tabled subgoal.
- ♦ **Consumer**: nodes that consume answers from the table space.
- ♦ **Interior**: nodes that are evaluated by the standard resolution.

Basic Tabling Definitions

- **Main Operations**
  - **Tabled Subgoal Call:** checks if the subgoal is in the table, and if not, adds a new entry for it and allocates a new generator node. Otherwise, it allocates a consumer node and starts consuming the available answers.

Basic Tabling Definitions

Main Operations

- **Tabled Subgoal Call:** checks if the subgoal is in the table, and if not, adds a new entry for it and allocates a new generator node. Otherwise, it allocates a consumer node and starts consuming the available answers.
- **New Answer:** verifies whether a newly found answer is already in the table, and if not, inserts it.

Basic Tabling Definitions

➤ Main Operations

- ♦ **Tabled Subgoal Call:** checks if the subgoal is in the table, and if not, adds a new entry for it and allocates a new generator node. Otherwise, it allocates a consumer node and starts consuming the available answers.
- ♦ **New Answer:** verifies whether a newly found answer is already in the table, and if not, inserts it.
- ♦ **Answer Resolution:** is executed everytime the computation reaches a consumer. It consumes the next available answer, and if not any, it **suspends** the current computation and schedules a possible resolution to continue execution.

Basic Tabling Definitions

➤ Main Operations

- ♦ **Tabled Subgoal Call:** checks if the subgoal is in the table, and if not, adds a new entry for it and allocates a new generator node. Otherwise, it allocates a consumer node and starts consuming the available answers.
- ♦ **New Answer:** verifies whether a newly found answer is already in the table, and if not, inserts it.
- ♦ **Answer Resolution:** is executed everytime the computation reaches a consumer. It consumes the next available answer, and if not any, it **suspends** the current computation and schedules a possible resolution to continue execution.
- ♦ **Completion:** is executed everytime the computation reaches a generator without untried alternatives. If the subgoal is **completely evaluated** it closes its table entry and reclaims space. Otherwise, it resumes one of the consumers with unconsumed answers.

Basic Tabling Definitions

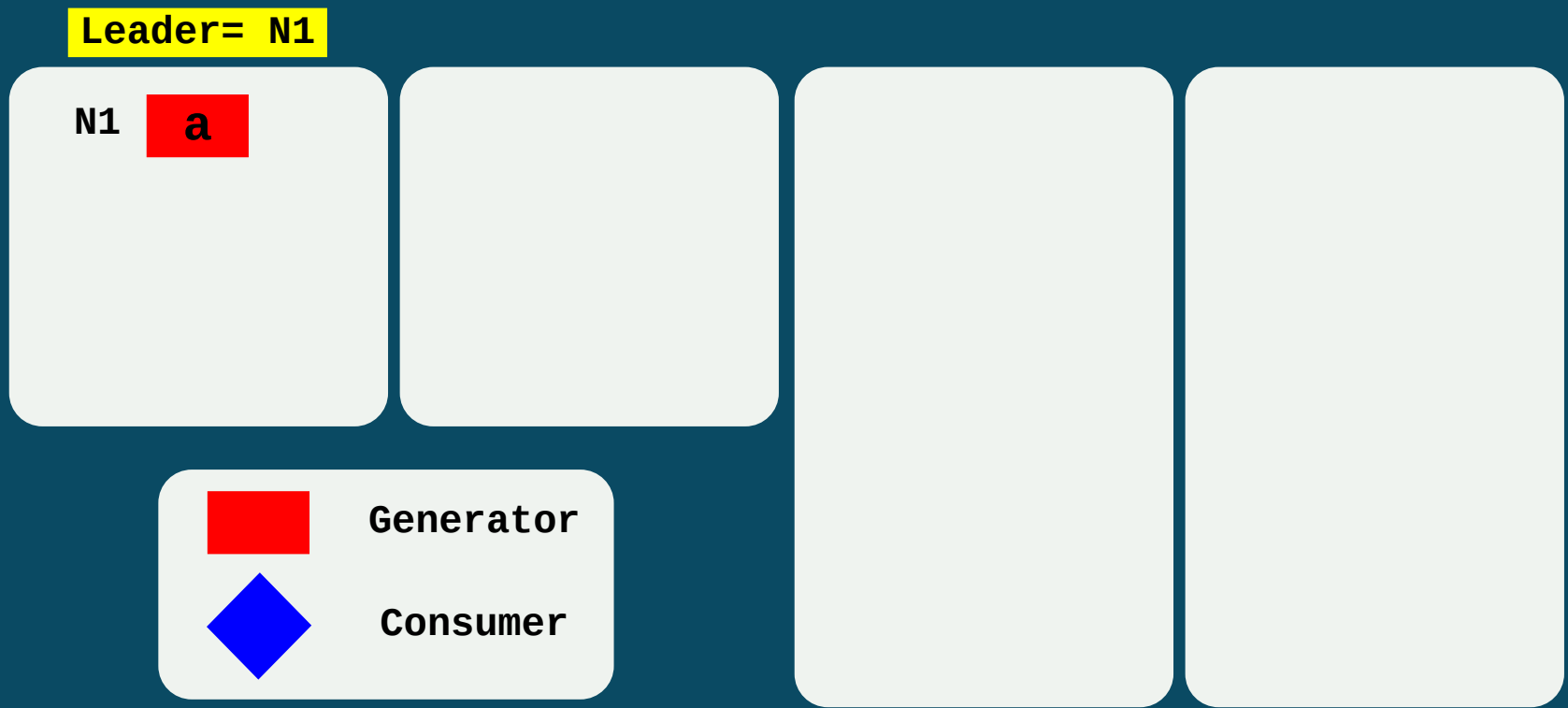
Leader Nodes

- A number of generators may be mutually dependent (**strongly connected component**) and thus they can only be completed together.
- The youngest generator which does not depend on older generators is called the **leader node**. A leader node defines the current completion point.

Basic Tabling Definitions

➤ Leader Nodes

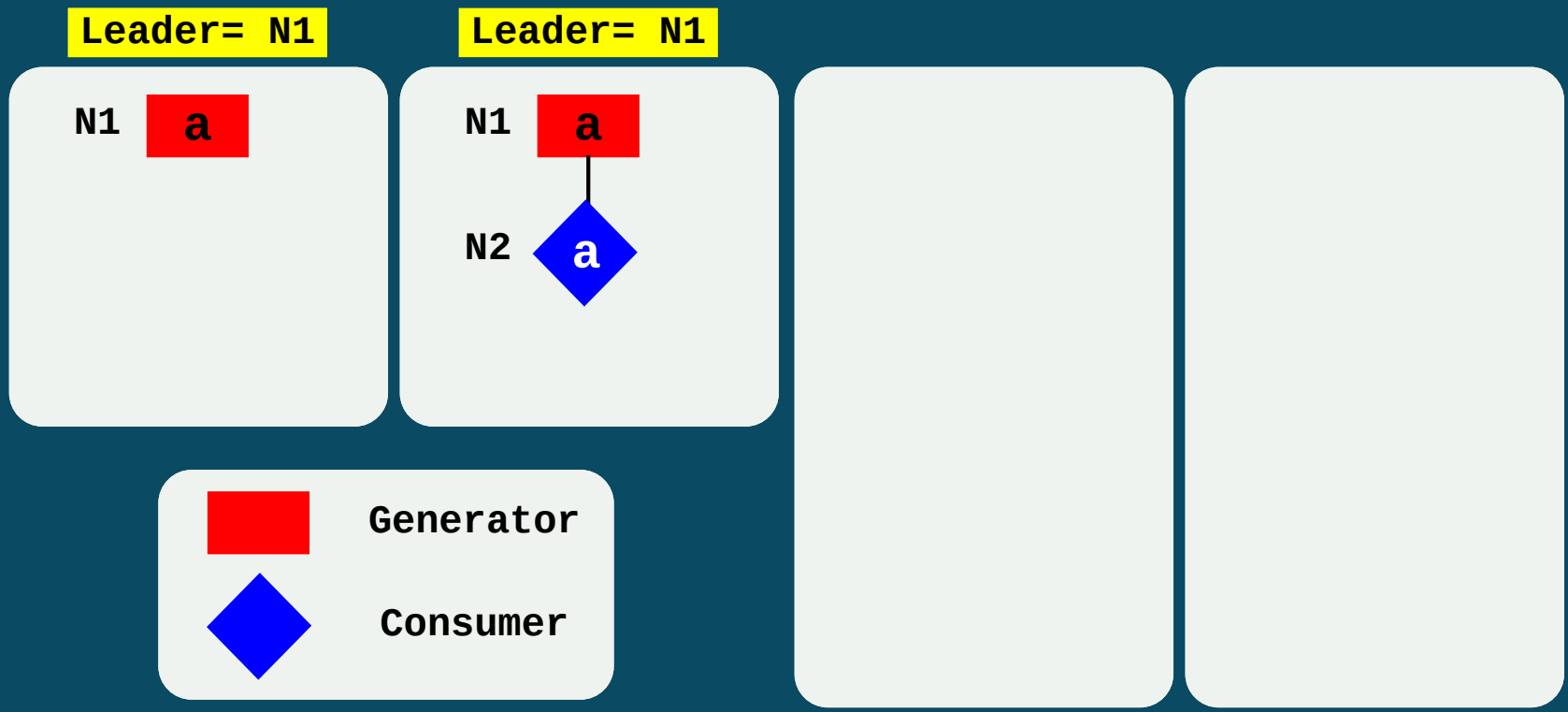
- ♦ A number of generators may be mutually dependent (**strongly connected component**) and thus they can only be completed together.
- ♦ The youngest generator which does not depend on older generators is called the **leader node**. A leader node defines the current completion point.



Basic Tabling Definitions

➤ Leader Nodes

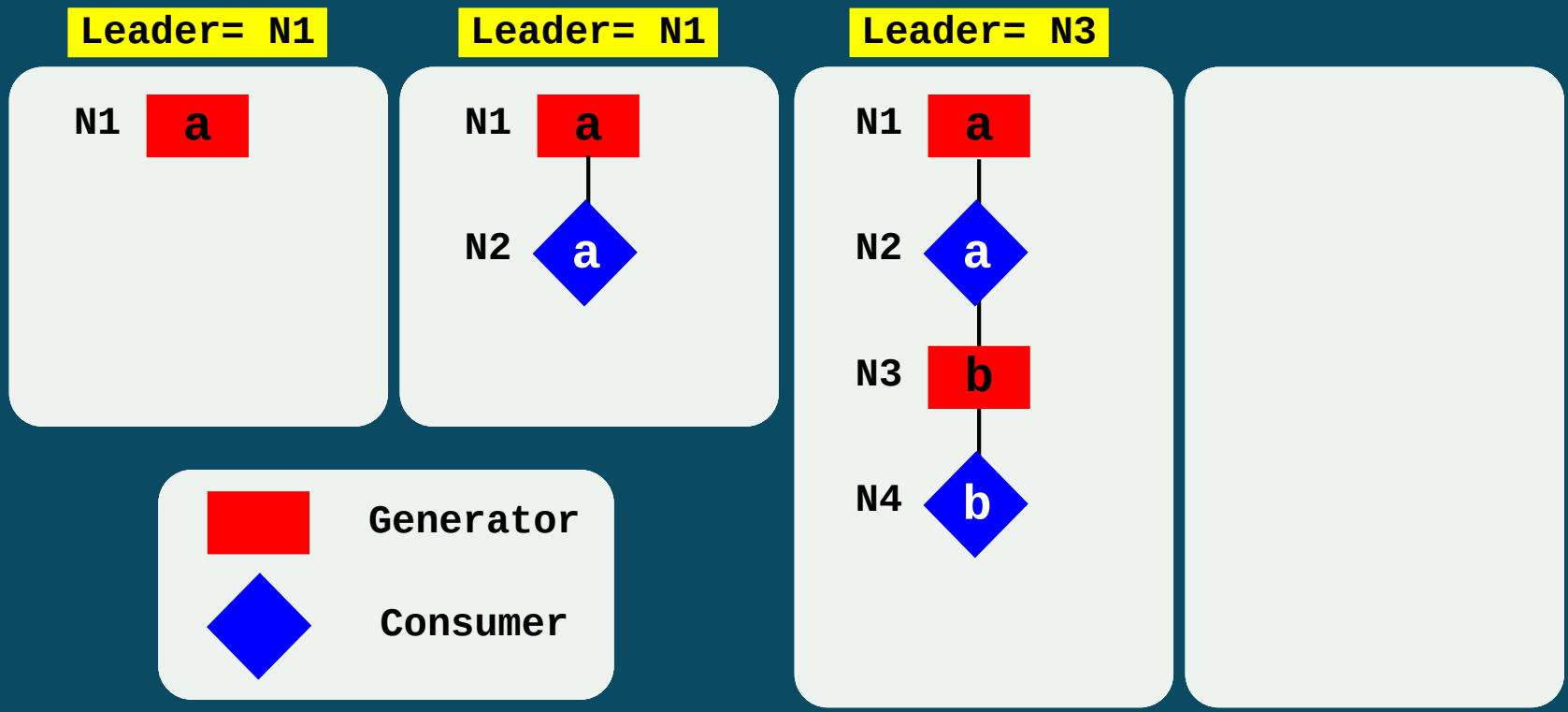
- ♦ A number of generators may be mutually dependent (**strongly connected component**) and thus they can only be completed together.
- ♦ The youngest generator which does not depend on older generators is called the **leader node**. A leader node defines the current completion point.



Basic Tabling Definitions

➤ Leader Nodes

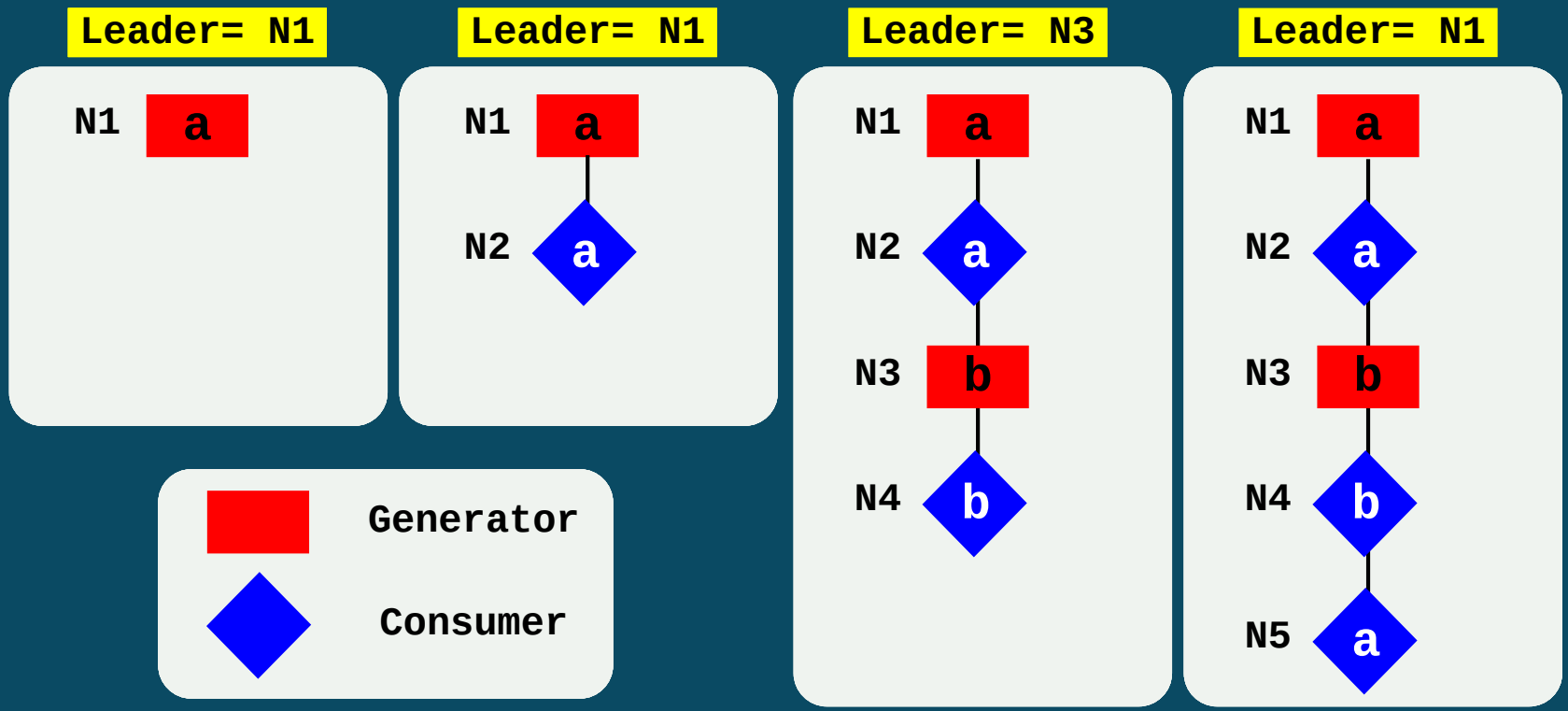
- ♦ A number of generators may be mutually dependent (**strongly connected component**) and thus they can only be completed together.
- ♦ The youngest generator which does not depend on older generators is called the **leader node**. A leader node defines the current completion point.



Basic Tabling Definitions

➤ Leader Nodes

- ♦ A number of generators may be mutually dependent (**strongly connected component**) and thus they can only be completed together.
- ♦ The youngest generator which does not depend on older generators is called the **leader node**. A leader node defines the current completion point.

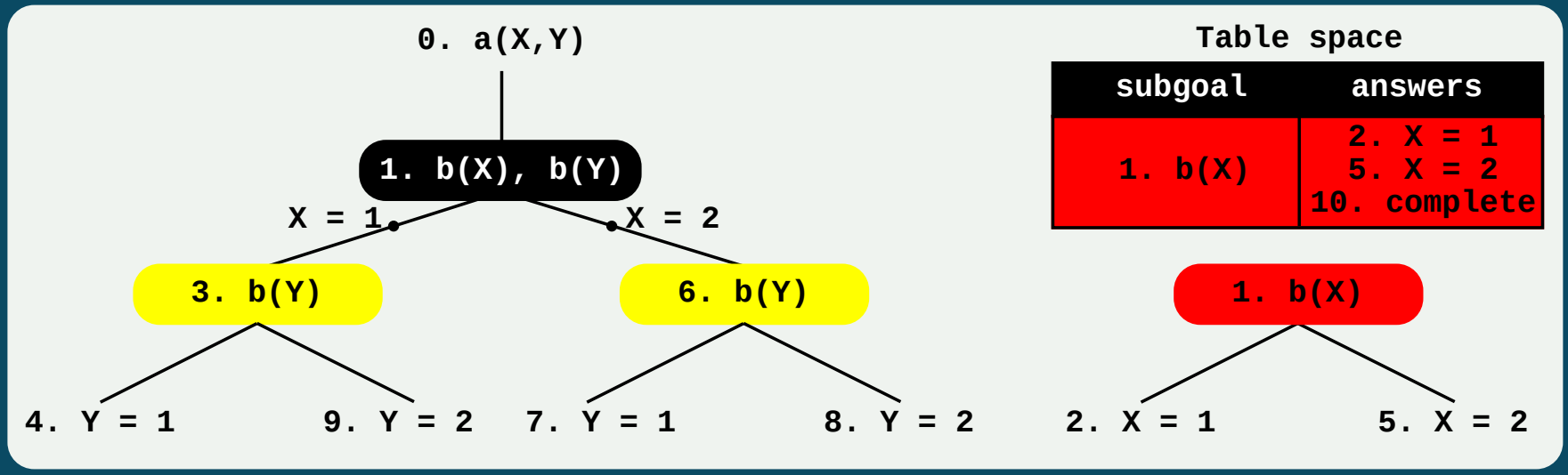


Batched Scheduling

- **Batched Scheduling:** tries to minimize the need to move around the search tree by batching the return of answers. When new answers are found, they are added to the table and the evaluation continues. Only later, when checking for completion, these answers are returned to the consumers.

```
:- table b/1.
a(X,Y) :- b(X), b(Y).
b(1).
b(2).

?- a(x,Y).
```



Local Scheduling

- **Local Scheduling:** tries to complete subgoals sooner. When new answers are found, they are added to the table and the evaluation fails. Answers are only returned when all alternatives for the subgoal in hand were tried.

```
:- table b/1.
a(X,Y) :- b(X), b(Y).
b(1).
b(2).

?- a(x,Y).
```

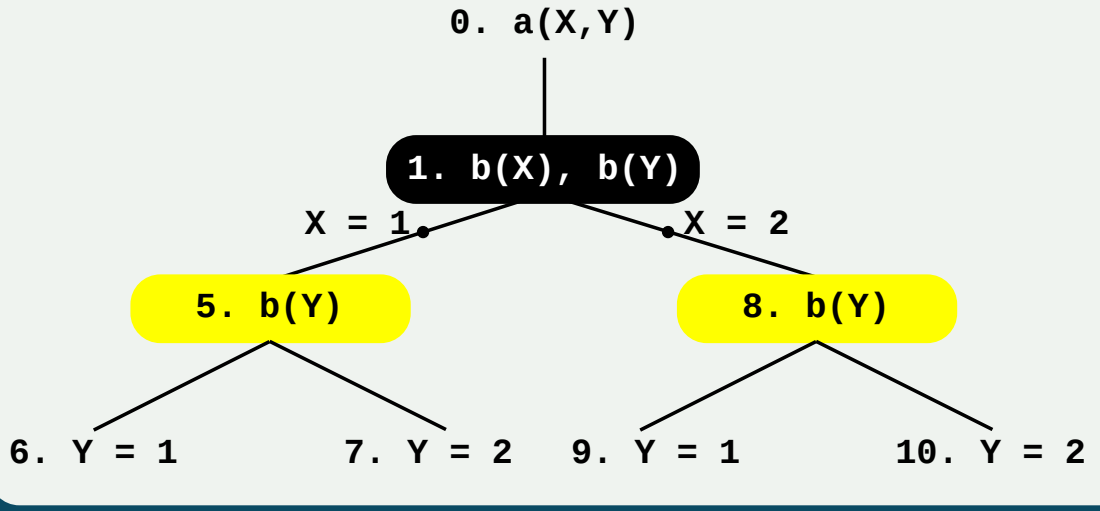
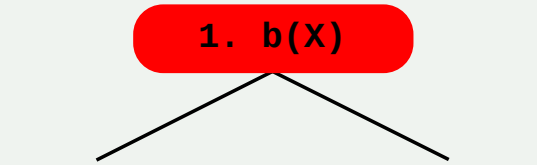


Table space

subgoal	answers
1. b(X)	2. X = 1 3. X = 2 4. complete



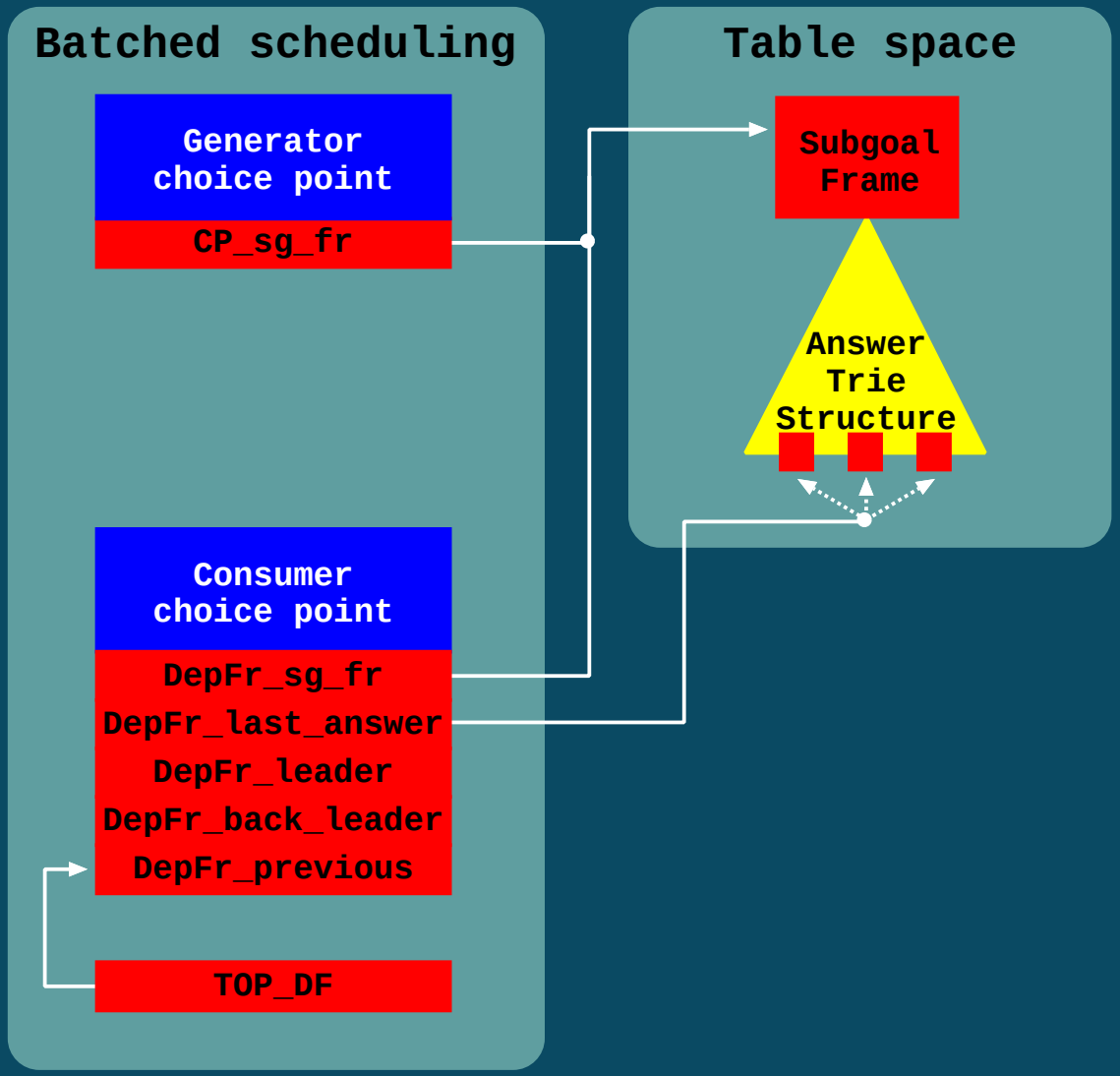
Batched x Local Scheduling

- **New Answers**
  - In batched when a new answer is found, the evaluation continues. In local, the evaluation fails.
- **Generators**
  - In batched a generator without untried alternatives, checks for completion. In local, the generator as also to act like a consumer and start consuming the set of found answers.

Batched x Local Scheduling

- **New Answers**
  - In batched when a new answer is found, the evaluation continues. In local, the evaluation fails.
- **Generators**
  - In batched a generator without untried alternatives, checks for completion. In local, the generator as also to act like a consumer and start consuming the set of found answers.
- **Our Approach**
  - Extend consumer nodes with a new data structure, the **dependency frame**.
  - Consider the whole set of consumers as a single group.
  - Avoid a separate completion stack.

Tabled Nodes for Batched Scheduling



Completion for Batched Scheduling

GN is the current leader node ?

- **No** → Backtrack
- **Yes** → Younger consumer node CN with unconsumed answers ?
  - **Yes** → Resume computation to CN
  - **No** → Perform completion
    . → Backtrack

Completion for Batched Scheduling

GN is the current leader node ?
GN is younger than TOP_DF Or DepFr_leader(TOP_DF) == GN

- **No** → Backtrack
- **Yes** → Younger consumer node CN with unconsumed answers ?
  - **Yes** → Resume computation to CN
    . **DepFr_back_leader(CN) = GN**
  - **No** → Perform completion
    . → Backtrack

Answer Resolution

Unconsumed answers in CN?

- **Yes** → Load next answer and proceed
- **No** → First time here ?

  - **Yes** → Backtrack
  - **No** → Resume computation to the younger node of
    - Last leader where completion was executed

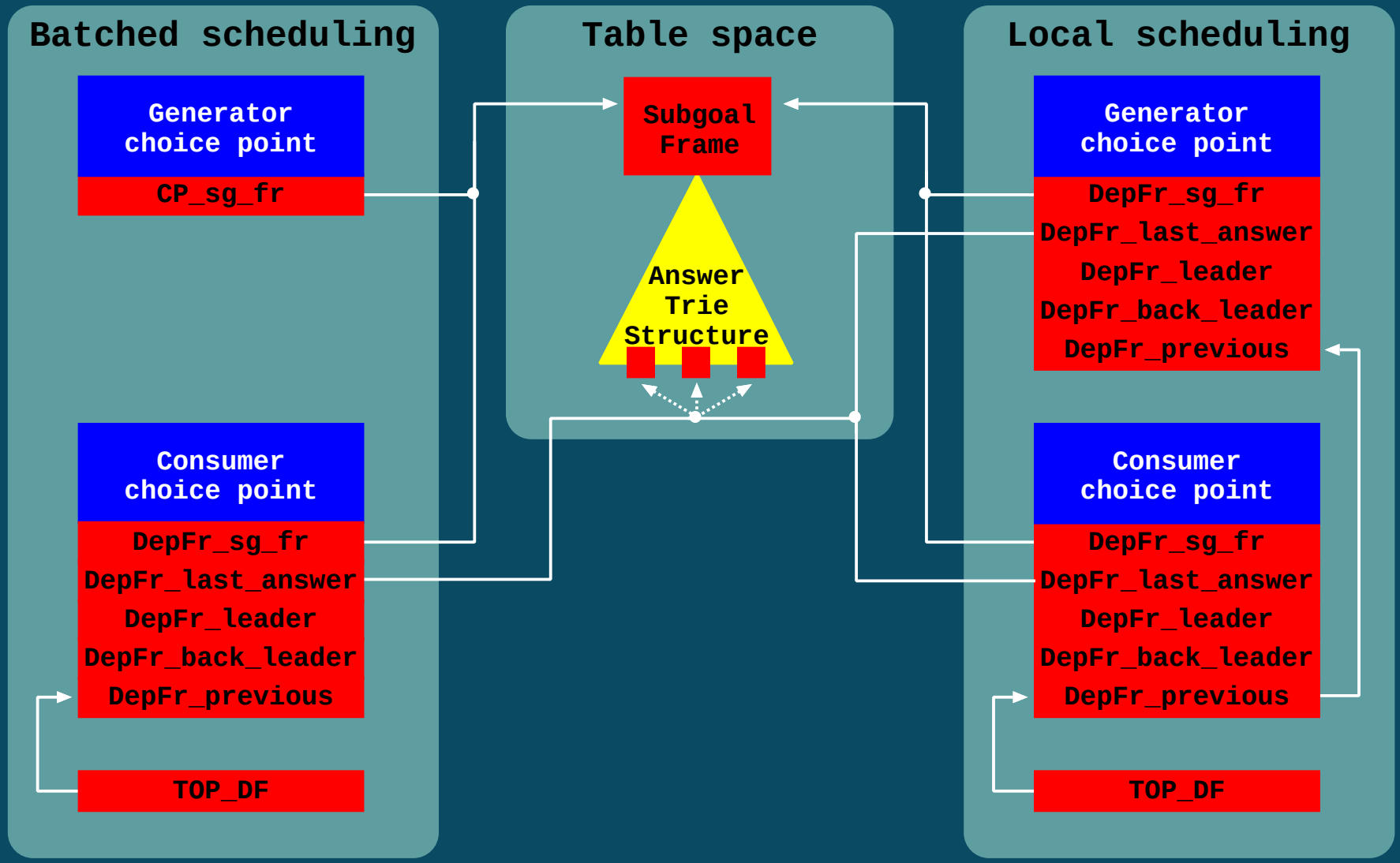
    - Previous consumer PN with unconsumed answers

Answer Resolution

Unconsumed answers in CN?

- **Yes** → Load next answer and proceed
- **No** → First time here ?
  . **DepFr_back_leader(CN) == NULL**
  - **Yes** → Backtrack
  - **No** → Resume computation to the younger node of
    - Last leader where completion was executed
      DepFr_back_leader(CN)
    - Previous consumer PN with unconsumed answers
      DepFr_back_leader(PN) = DepFr_back_leader(CN)

Tabled Nodes for Local Scheduling



Completion for Local Scheduling

GN is the current leader node ?

- **No** → Load first answer and proceed
- **Yes** → Younger consumer node CN with unconsumed answers ?
  - **Yes** → Resume computation to CN
  - **No** → Perform completion
    . → Load first answer and proceed

Completion for Local Scheduling

GN is the current leader node ?
DepFr_leader(TOP_DF) == GN

- **No** → Load first answer and proceed
  . **CP_alt(GN) = answer_resolution**
- **Yes** → Younger consumer node CN with unconsumed answers ?
  - **Yes** → Resume computation to CN
    . **DepFr_back_leader(CN) = GN**
  - **No** → Perform completion
    . → Load first answer and proceed
    . **CP_alt(GN) = answer_resolution**

Discussion

- Currently, we have already batched and local scheduling functioning separately.

Discussion

- Currently, we have already batched and local scheduling functioning separately.
- To support mixed-strategy only two new features have to be addressed:
  - Support strategy-specific Prolog declarations like **:- batched path/2.** and **:- local path/2.** to define the strategy to be used for a predicate.
  - Accordingly to the declared strategy generate appropriate tabling instructions, such as **batched_new_answer**/**local_new_answer** and **batched_completion**/**local_completion**.

Discussion

- Currently, we have already batched and local scheduling functioning separately.
- To support mixed-strategy only two new features have to be addressed:
  - Support strategy-specific Prolog declarations like **:- batched path/2.** and **:- local path/2.** to define the strategy to be used for a predicate.
  - Accordingly to the declared strategy generate appropriate tabling instructions, such as **batched_new_answer**/**local_new_answer** and **batched_completion**/**local_completion**.
- YapTab can also be easily extended to support:
  - Dynamic switching of strategy at the predicate level.
  - Mixed-strategy evaluation at the subgoal level (per generator).
  - Dynamic switching from batched to local scheduling, while a generator is still producing new answers.

Further Work

- Adjust the system to support mixed-strategy evaluation.
- Use a set of common tabled benchmarks to investigate and study the impact of combining both strategies for tabled evaluation.