On Combining Linear-Based Strategies For Tabled Evaluation of Logic Programs

Miguel Areias and Ricardo Rocha
CRACS & INESC-Porto LA
Faculty of Sciences, University of Porto, Portugal
miguel-areias@dcc.fc.up.pt *ricroc@dcc.fc.up.pt*

ICLP 2011, Lexington , USA, July 2011

Tabling in Logic Programming

- Tabling is an implementation technique that overcomes some of the limitations of SLD resolution.
  - **Positive Infinite Cycles** (insufficient expressiveness)
  - **Negative Infinite Cycles** (inconsistence)
  - **Redundant Computations** (inefficiency)
- Implementations of tabling are currently available in systems like XSB Prolog, Yap Prolog, B-Prolog, ALS-Prolog, Mercury and more recently Ciao Prolog.

Tabling in Logic Programming

- Tabling is an implementation technique that overcomes some of the limitations of SLD resolution.
  - **Positive Infinite Cycles** (insufficient expressiveness)
  - **Negative Infinite Cycles** (inconsistence)
  - **Redundant Computations** (inefficiency)
- Implementations of tabling are currently available in systems like XSB Prolog, Yap Prolog, B-Prolog, ALS-Prolog, Mercury and more recently Ciao Prolog.
- In these implementations, we can distinguish two main categories of tabling mechanisms:
  - **Suspension-Based Tabling**: can be seen as a sequence of sub-computations that can be suspended and later resumed, when necessary, to compute fix-points (XSB Prolog, Yap Prolog, Mercury and Ciao Prolog).
  - **Linear Tabling**: can be seen as a single execution tree where tabled subgoals use iterative computations, without requiring suspension and resumption, to compute fix-points (B-Prolog, ALS Prolog and Yap Prolog).

Linear Tabling

- The two most well-known linear tabling strategies are:
  - **DRE (Dynamic Reordering of Execution)**: repeated calls, **the followers**, execute from the backtracking point of the former call. A follower is then repeatedly re-executed, until all the available answers and clauses have been exhausted, that is, until a fix-point is reached (B-Prolog and Yap Prolog).
  - **DRA (Dynamic Reordering of Alternatives)**: tables not only the answers to tabled subgoals, but also the alternatives leading to repeated calls, the **looping alternatives**. It then uses the looping alternatives to repeatedly recompute them until reaching a fix-point (ALS Prolog and Yap Prolog).

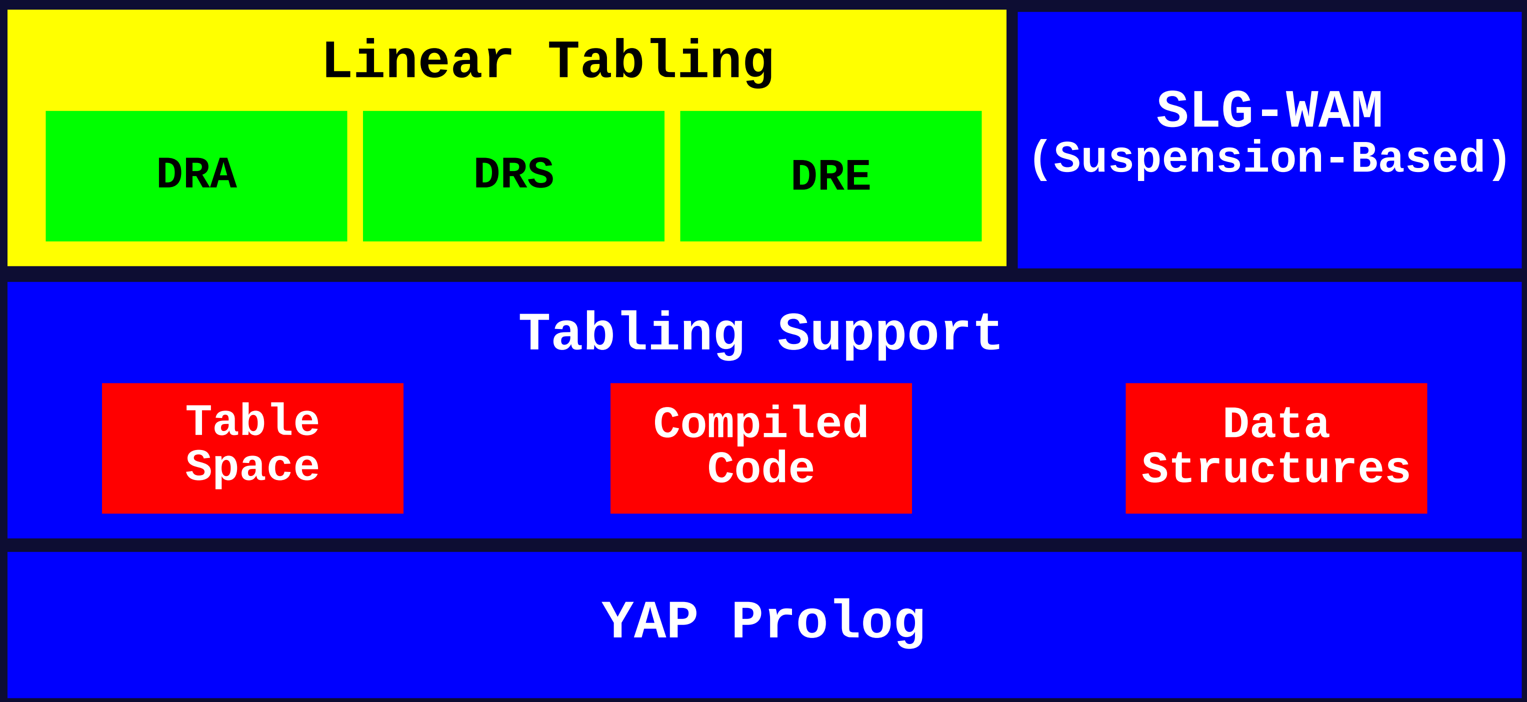
Linear Tabling

- The two most well-known linear tabling strategies are:
  - **DRE (Dynamic Reordering of Execution)**: repeated calls, **the followers**, execute from the backtracking point of the former call. A follower is then repeatedly re-executed, until all the available answers and clauses have been exhausted, that is, until a fix-point is reached (B-Prolog and Yap Prolog).
  - **DRA (Dynamic Reordering of Alternatives)**: tables not only the answers to tabled subgoals, but also the alternatives leading to repeated calls, the **looping alternatives**. It then uses the looping alternatives to repeatedly recompute them until reaching a fix-point (ALS Prolog and Yap Prolog).

- In this work, we also propose a new linear tabling strategy:
  - **DRS (Dynamic Reordering of Solutions)**: it can be seen as a variant of the DRA strategy, but applied to the consumption of solutions. The key idea is to memorize the solutions leading to consumer calls, the **looping solutions**, and use them as the DRA strategy uses the looping alternatives (Yap Prolog).

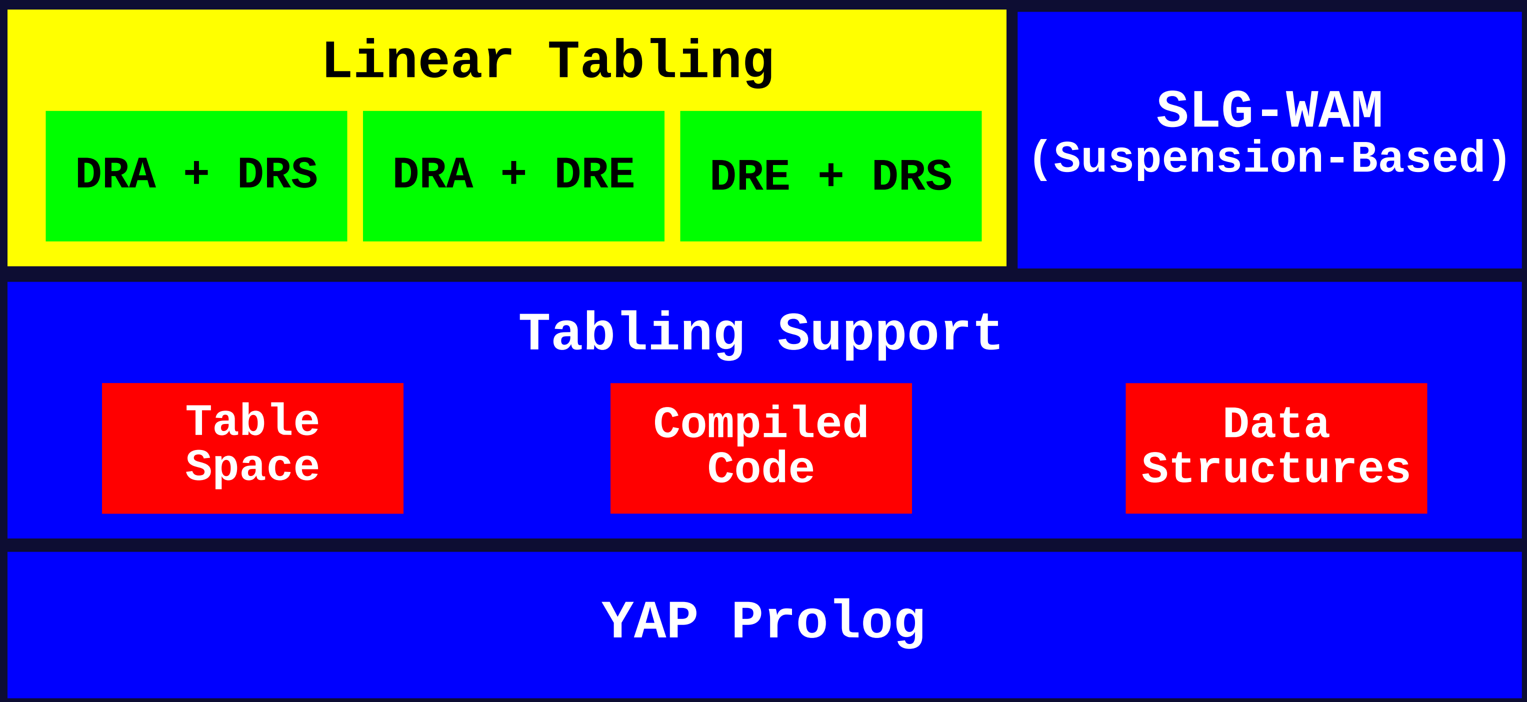
Our Goal

- Implement a framework on top of the **Yap Prolog system**, that supports the **combination of the three strategies**.
- Analyze the **advantages and weaknesses** of each strategy, when used **solely or combined** with the others.



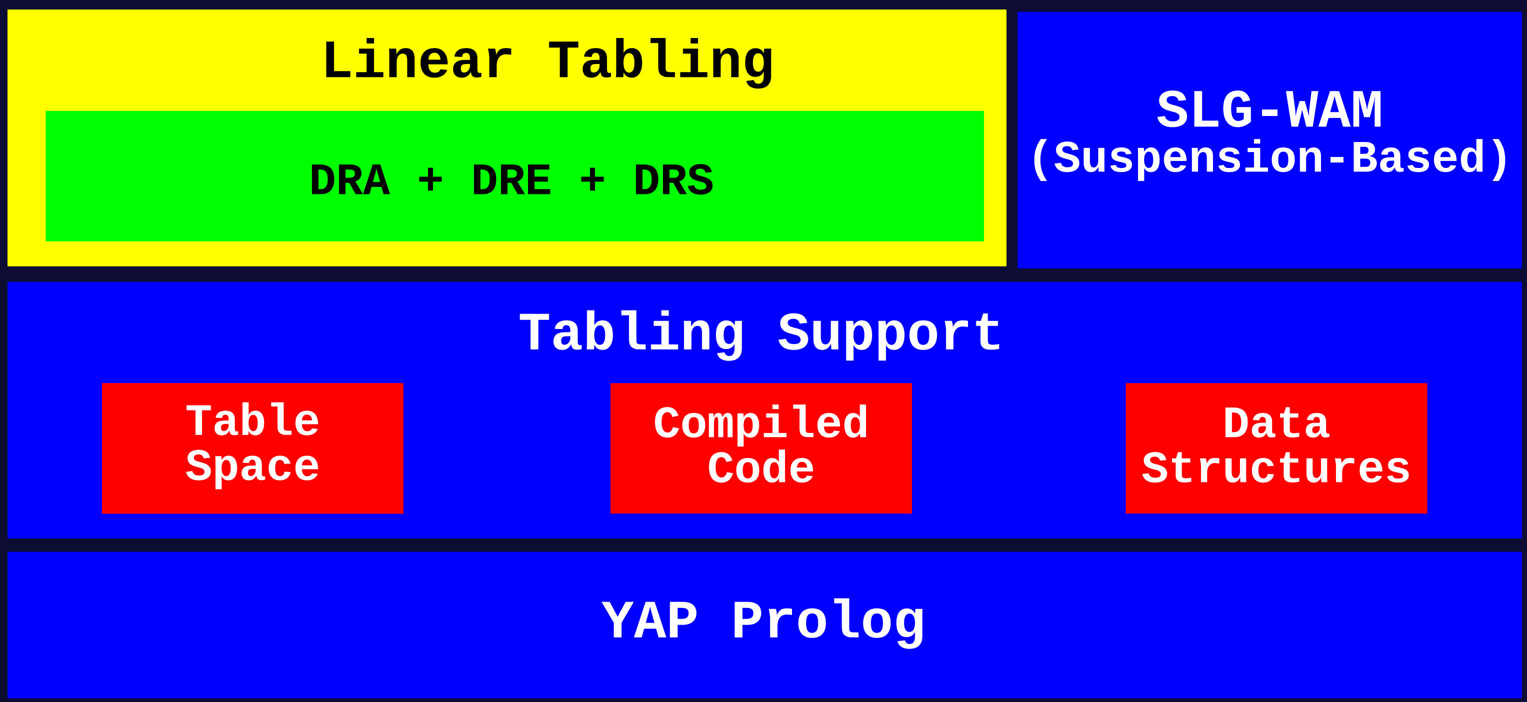
Our Goal

- Implement a framework on top of the **Yap Prolog system**, that supports the **combination of the three strategies**.
- Analyze the **advantages and weaknesses** of each strategy, when used **solely or combined** with the others.



Our Goal

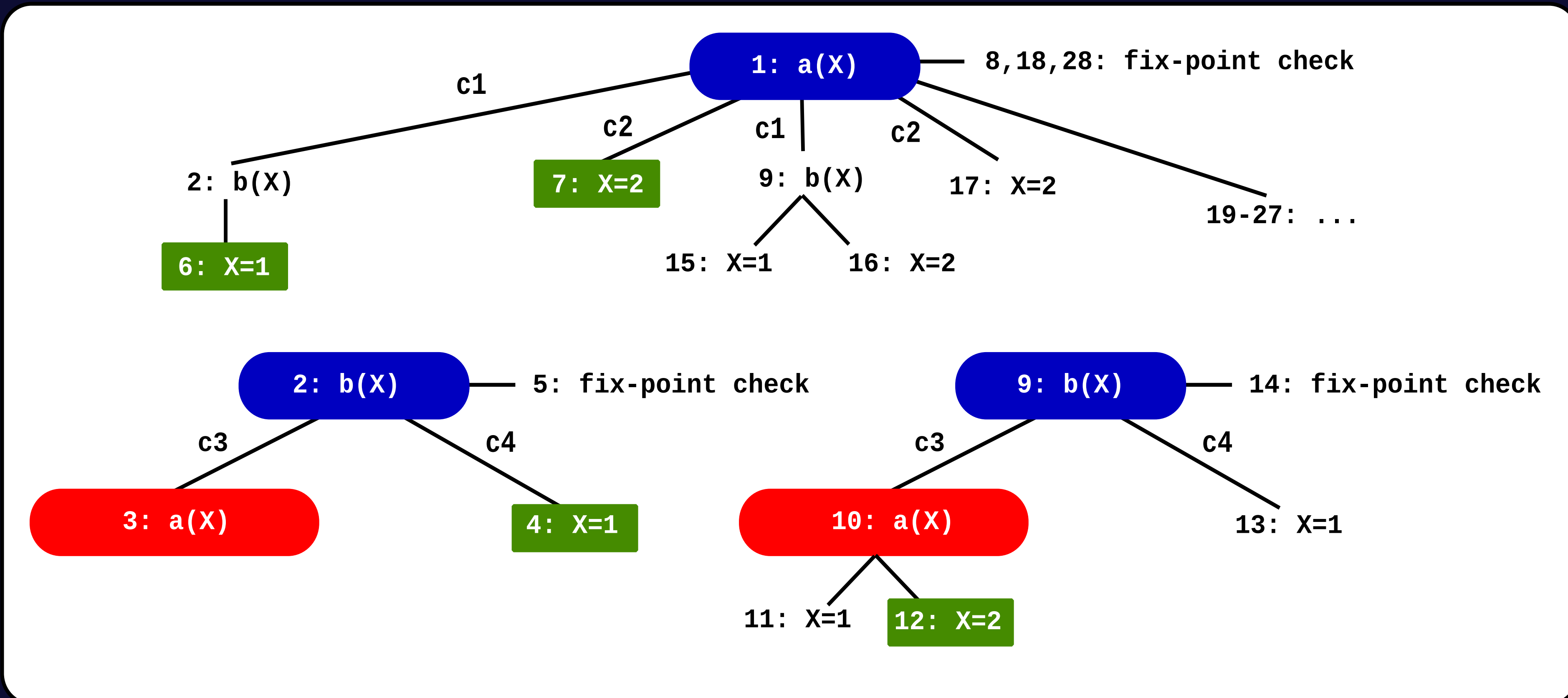
- Implement a framework on top of the **Yap Prolog system**, that supports the **combination of the three strategies**.
- Analyze the **advantages and weaknesses** of each strategy, when used **solely or combined** with the others.



Evaluation Example I - Standard Linear Tabling

```
:- table a/1, b/1.

a(X):- b(X).          (c1)
a(2).                 (c2)
b(X):- a(X).          (c3)
b(1).                 (c4)
```

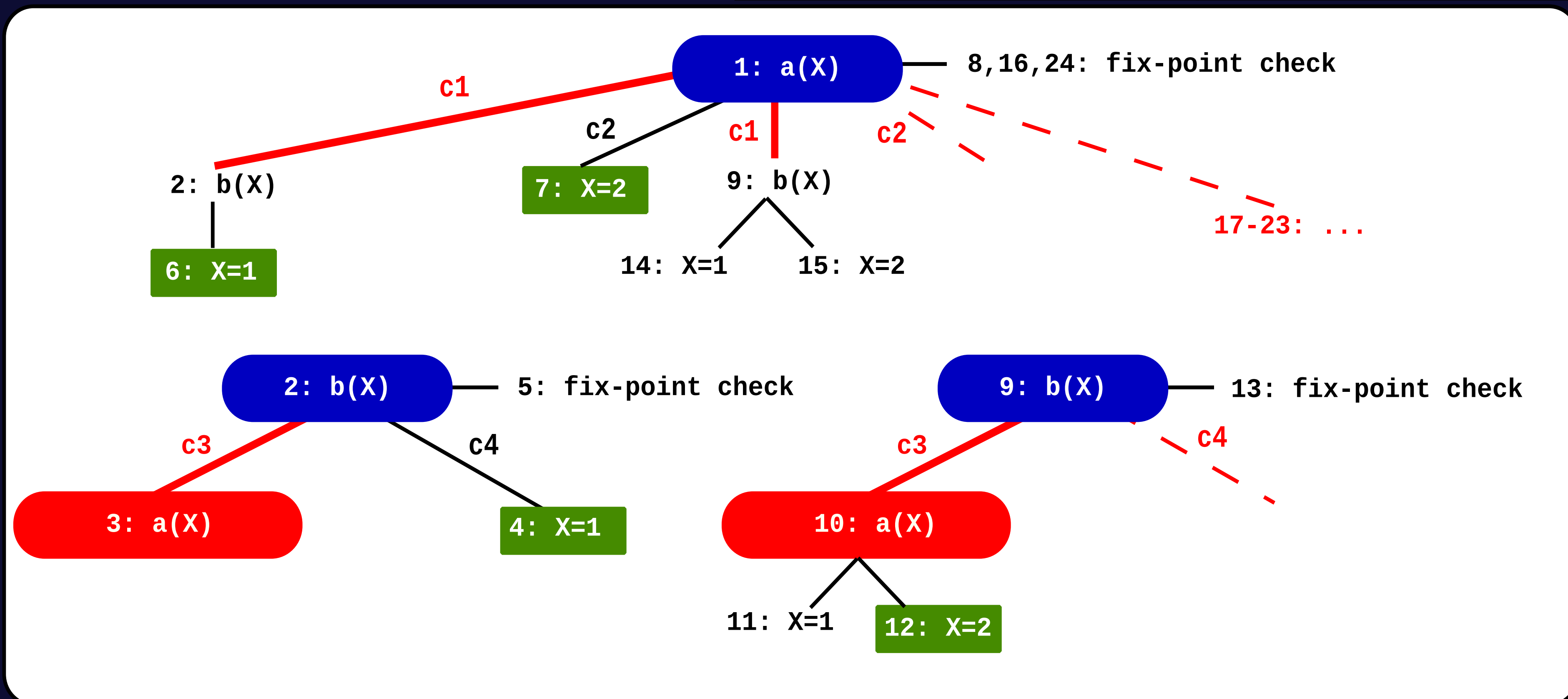
Call	Solutions
1: a(X)	6: X=1 7: X=2
2: b(X)	4: X=1 12: X=2



Evaluation Example I - DRA

```
:- table a/1, b/1.

a(X):- b(X).          (c1)
a(2).                 (c2)
b(X):- a(X).          (c3)
b(1).                 (c4)
```

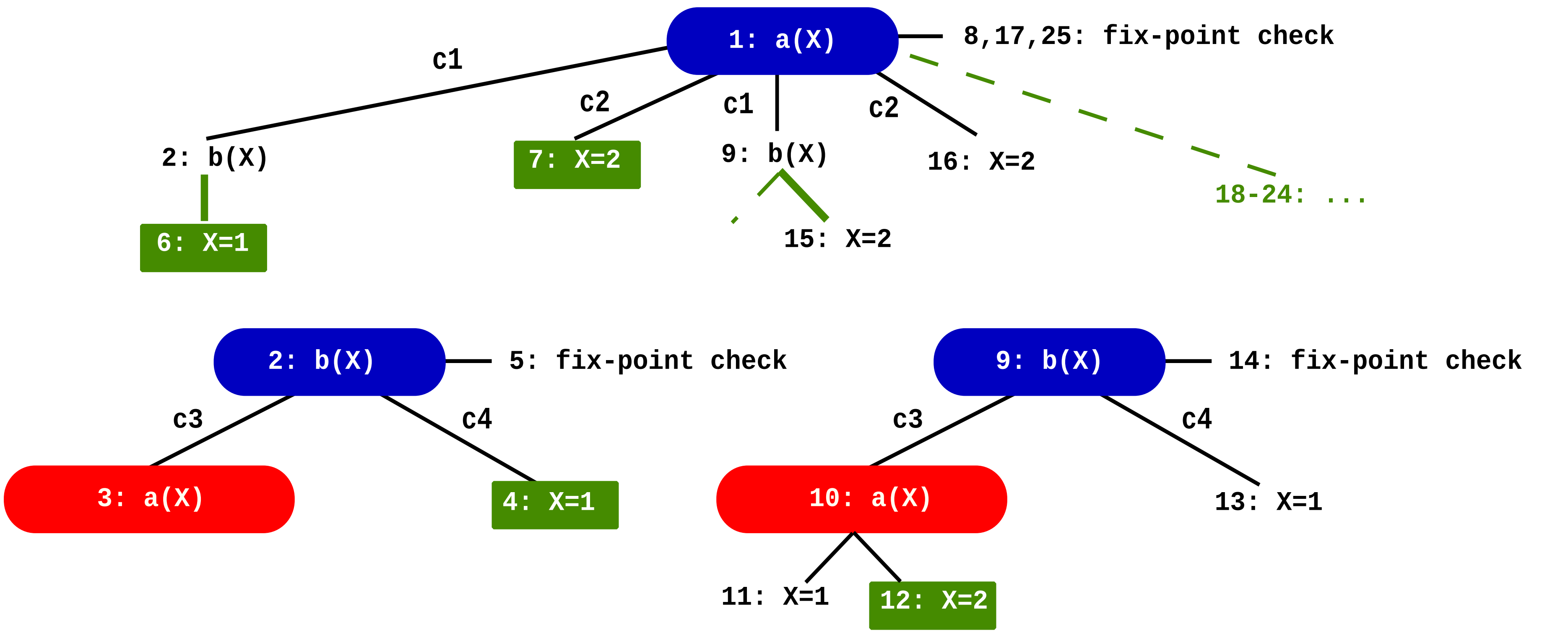
Call	Solutions	Looping Alternatives
1: a(X)	6: X=1 7: X=2	3: c1
2: b(X)	4: X=1 12: X=2	3: c3



Evaluation Example I - DRS

```
:- table a/1, b/1.

a(X):- b(X).              (c1)
a(2).                     (c2)
b(X):- a(X).              (c3)
b(1).                     (c4)
```

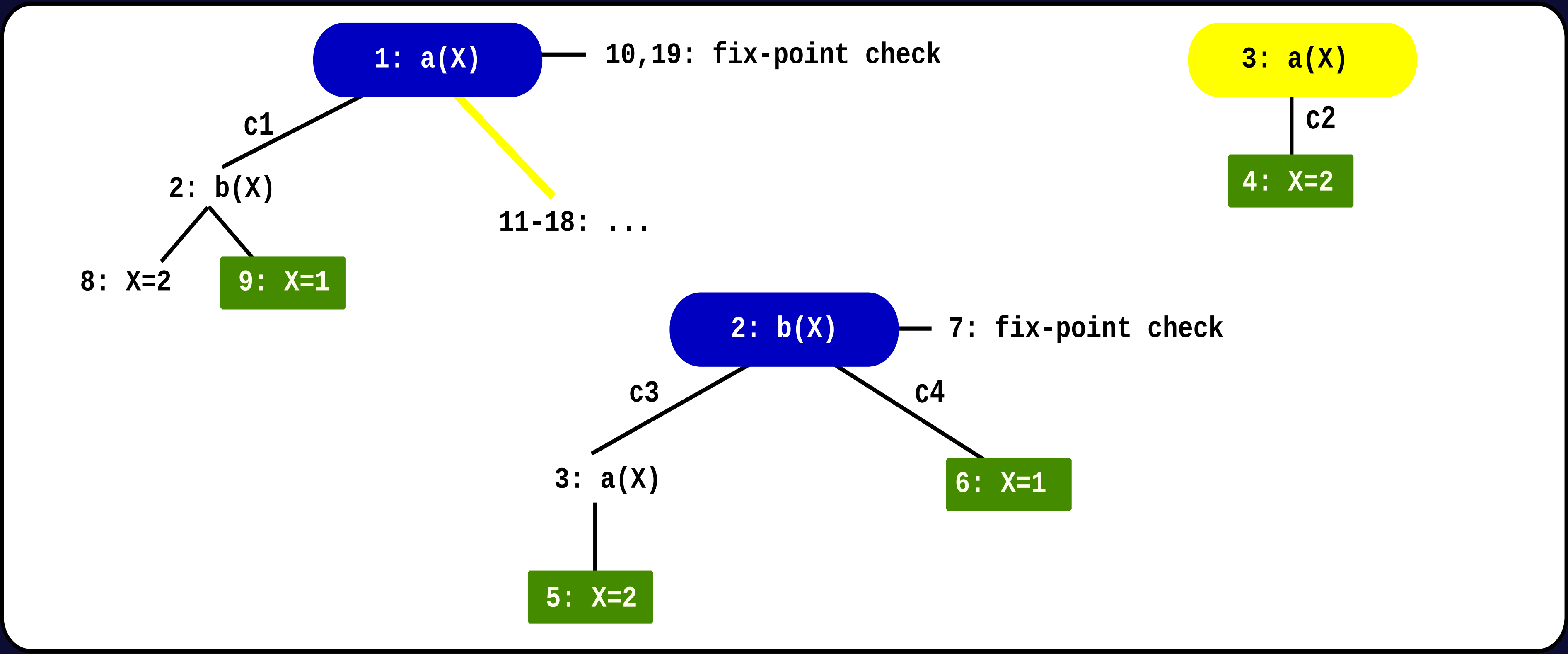
Call	Solutions	Looping Solutions
1: a(X)	6: X=1 7: X=2	
2: b(X)	4: X=1 12: X=2	



Evaluation Example I - DRE

```
:- table a/1, b/1.

a(X):- b(X).                (c1)
a(2).                       (c2)
b(X):- a(X).                (c3)
b(1).                       (c4)
```

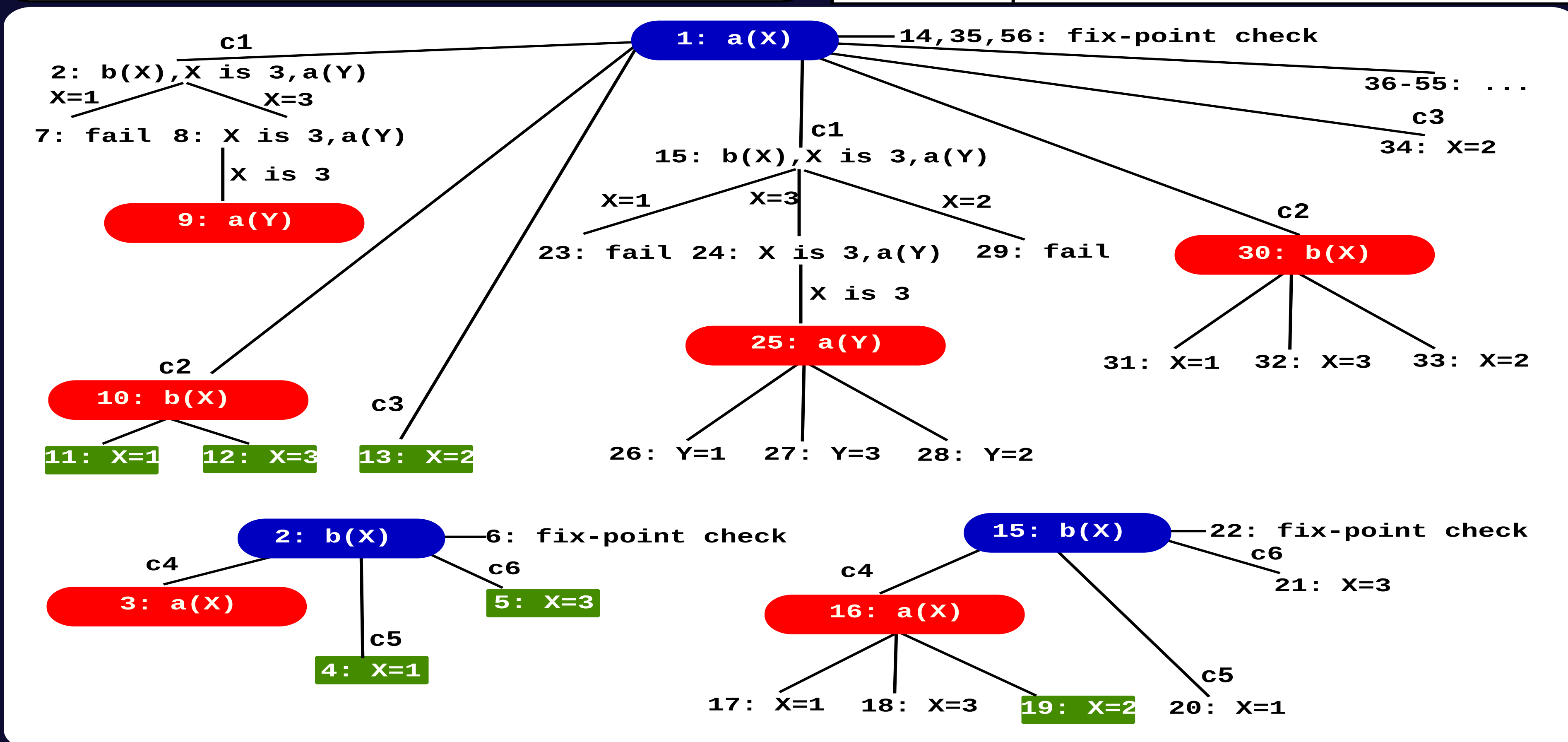
Call	Solutions
1: a(X)	4: X=2 9: X=1
2: b(X)	5: X=2 6: X=1



Evaluation Example II - Standard Linear Tabling

```
:-table a/1,b/1.
a(X):-b(X),X is 3,a(Y).     (c1)
a(X):-b(X).                 (c2)
a(2).                       (c3)
b(X):-a(X).                 (c4)
b(1).                       (c5)
b(3).                       (c6)
```

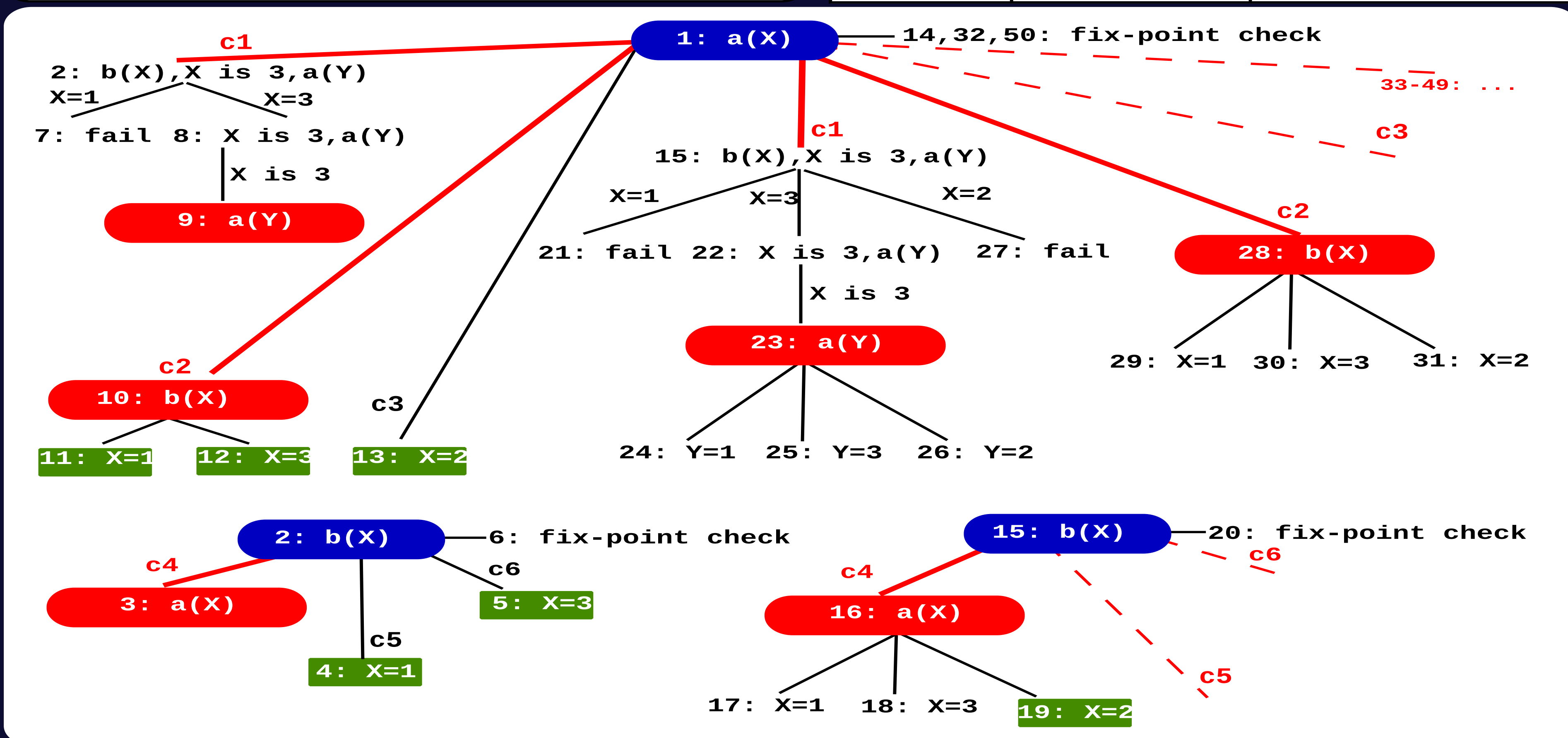
Call	Solutions
1: a(X)	11: X=1 12: X=3 13: X=2
2: b(X)	4: X=1 5: X=3 19: X=2



Evaluation Example II - DRA

```
:-table a/1,b/1.
a(X):-b(X),X is 3,a(Y).                    (c1)
a(X):-b(X).                                (c2)
a(2).                                      (c3)
b(X):-a(X).                                (c4)
b(1).                                      (c5)
b(3).                                      (c6)
```

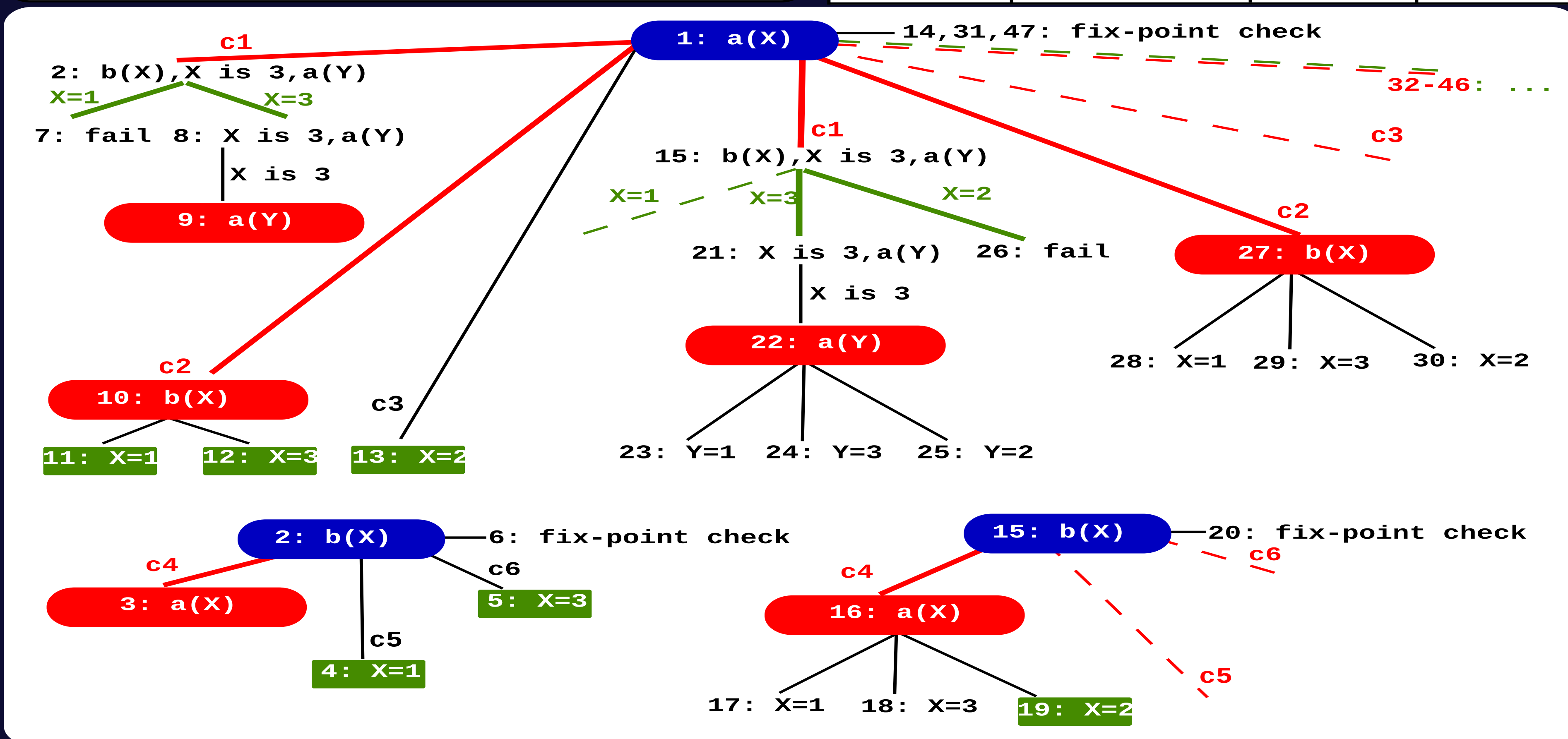
Call	Solutions	Loop Alt
1: a(X)	11: X=1 12: X=3 13: X=2	3: c1 10: c2
2: b(X)	4: X=1 5: X=3 19: X=2	3: c4



Evaluation Example II - DRA + DRS

```
:-table a/1,b/1.
a(X):-b(X),X is 3,a(Y).     (c1)
a(X):-b(X).                 (c2)
a(2).                       (c3)
b(X):-a(X).                 (c4)
b(1).                       (c5)
b(3).                       (c6)
```

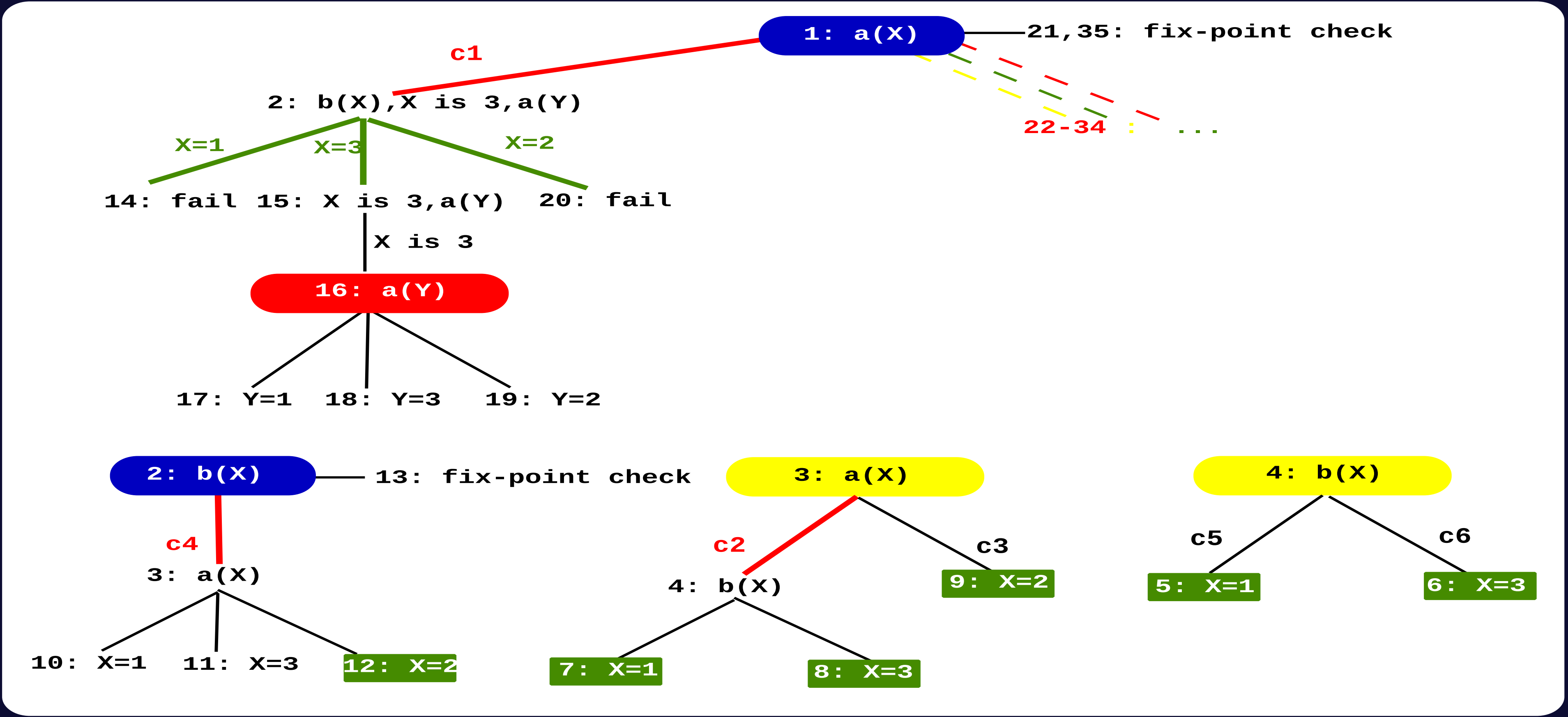
Call	Solutions	Loop Alt	Loop Sol
1: a(X)	11: X=1 12: X=3 13: X=2	3: c1 10: c2	
2: b(X)	4: X=1 5: X=3 19: X=2	3: c4	9: X=3



Evaluation Example II - DRA + DRS + DRE

```
:-table a/1,b/1.
a(X):-b(X),X is 3,a(Y).                         (c1)
a(X):-b(X).                                     (c2)
a(2).                                           (c3)
b(X):-a(X).                                     (c4)
b(1).                                           (c5)
b(3).                                           (c6)
```

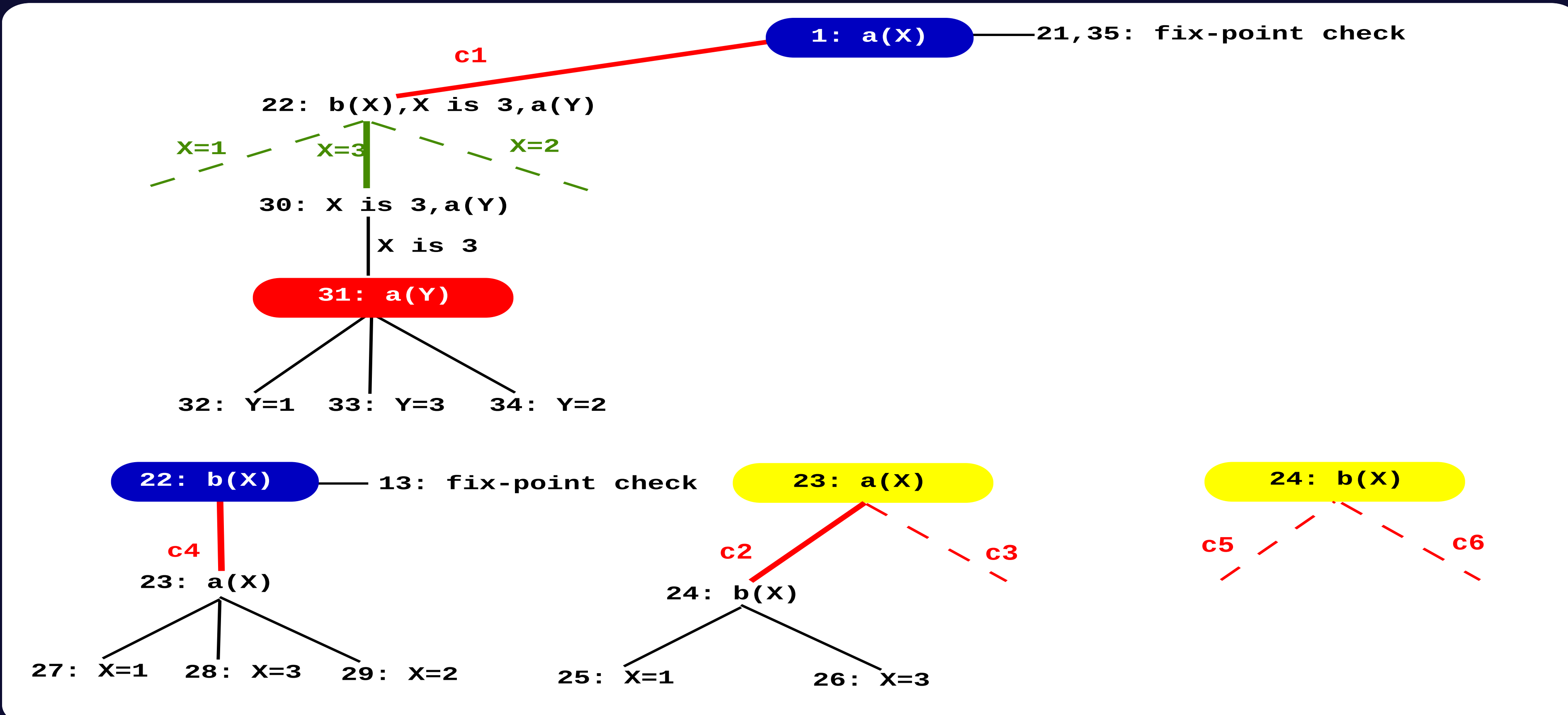
Call	Solutions	Loop Alt	Loop Sol
1: a(X)	7: X=1 8: X=3 9: X=2	3: c1 4: c2	
2: b(X)	5: X=1 6: X=3 12: X=2	3: c4	16: X=3



Evaluation Example II - DRA + DRS + DRE

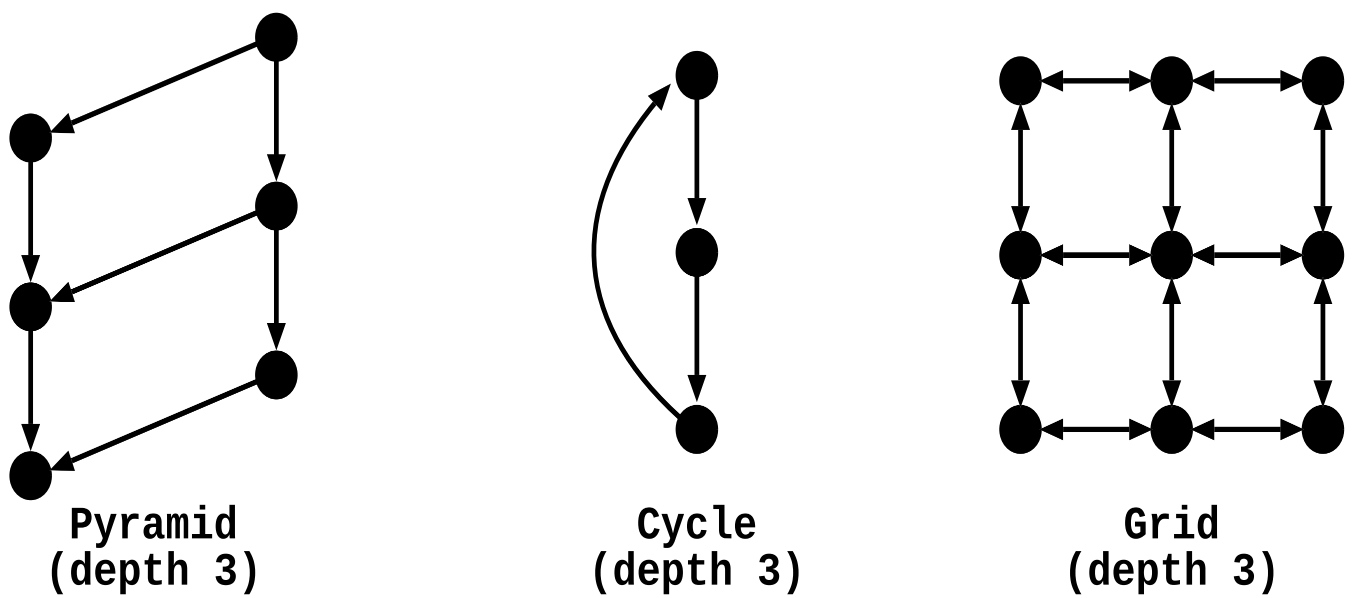
```
:-table a/1,b/1.
a(X):-b(X),X is 3,a(Y).          (c1)
a(X):-b(X).                      (c2)
a(2).                            (c3)
b(X):-a(X).                      (c4)
b(1).                            (c5)
b(3).                            (c6)
```

Call	Solutions	Loop Alt	Loop Sol
1: a(X)	7: X=1 8: X=3 9: X=2	3: c1 4: c2	
2: b(X)	5: X=1 6: X=3 12: X=2	3: c4	16: X=3



Path Problem - Configuration

```
path_first(X,Z) :- edge(X,Y) , path_first(Y,Z).
path_first(X,Z) :- edge(X,Z).

path_last(X,Z)  :- edge(X,Z).
path_last(X,Z)  :- edge(X,Y) , path_last(Y,Z).
```



Experimental Results

Strategy	Pyramid			Cycle			Grid		
	1000	2000	3000	1000	2000	3000	20	30	40
Recursive Clause First									
Standard	664	2,669	6,040	377	1,522	3,400	386	2,714	10,689
DRE	**1.02**	**1.01**	**1.02**	1.00	**1.01**	**1.01**	**1.02**	1.00	1.00
DRA	**1.55**	**1.51**	**1.51**	**1.22**	**1.23**	**1.21**	**1.14**	**1.09**	**1.10**
DRS	**1.01**	1.00	**1.01**	**1.21**	**1.23**	**1.22**	**1.23**	**1.27**	**1.31**
DRE+DRA	1.52	1.51	1.50	**1.24**	1.23	1.20	**1.15**	**1.10**	1.06
DRE+DRS	1.01	1.01	1.00	**1.22**	1.23	1.22	1.22	1.23	1.23
DRA+DRS	1.54	**1.52**	1.51	**1.56**	**1.57**	**1.52**	**1.42**	**1.42**	**1.43**
All	**1.56**	**1.53**	1.50	1.55	1.57	1.52	1.38	1.39	1.37
Recursive Clause Last									
Standard	673	2,775	6,216	382	1,542	3,487	365	2,602	10,403
DRE	0.99	**1.01**	**1.01**	**1.01**	**1.01**	**1.01**	**1.02**	**1.03**	**1.03**
DRA	**1.47**	**1.49**	**1.47**	**1.24**	**1.22**	**1.22**	**1.15**	**1.13**	**1.11**
DRS	0.99	0.99	**1.01**	**1.20**	**1.21**	**1.23**	**1.21**	**1.27**	**1.30**
DRE+DRA	**1.49**	1.34	1.43	1.24	1.22	1.22	1.14	1.12	1.10
DRE+DRS	1.00	0.99	1.01	**1.23**	1.22	1.23	**1.22**	1.27	1.30
DRA+DRS	1.47	1.47	1.46	**1.55**	**1.54**	**1.53**	**1.42**	**1.43**	**1.43**
All	1.49	1.48	1.09	1.48	**1.56**	**1.55**	1.42	**1.44**	**1.45**

Statistics For The Edge Grid 40

```
path_first(X,Z) :- sld1 , edge(X,Y) , path_first(Y,Z) , sld2.
path_first(X,Z) :- sld3 , edge(X,Z) , sld4.

path_last(X,Z)  :- ...
path_last(X,Z)  :- ...
```

Strategy	#SLD Computations			
	sld1/0	sld2/0	sld3/0	sld4/0
Recursive Clause First				
Standard	35,202	200,974,309	35,201	149,757
DRE	**1.05**	**1.04**	**1.05**	**1.04**
DRA	1.00	**1.05**	**21.99**	**12.00**
DRS	1.00	**1.29**	1.00	1.00
DRE+DRA	1.05	**1.10**	1.07	1.11
DRE+DRS	1.05	**1.33**	1.05	1.04
DRA+DRS	1.00	**1.38**	21.99	12.00
All	1.05	**1.43**	1.07	1.11

Statistics For The Edge Grid 40

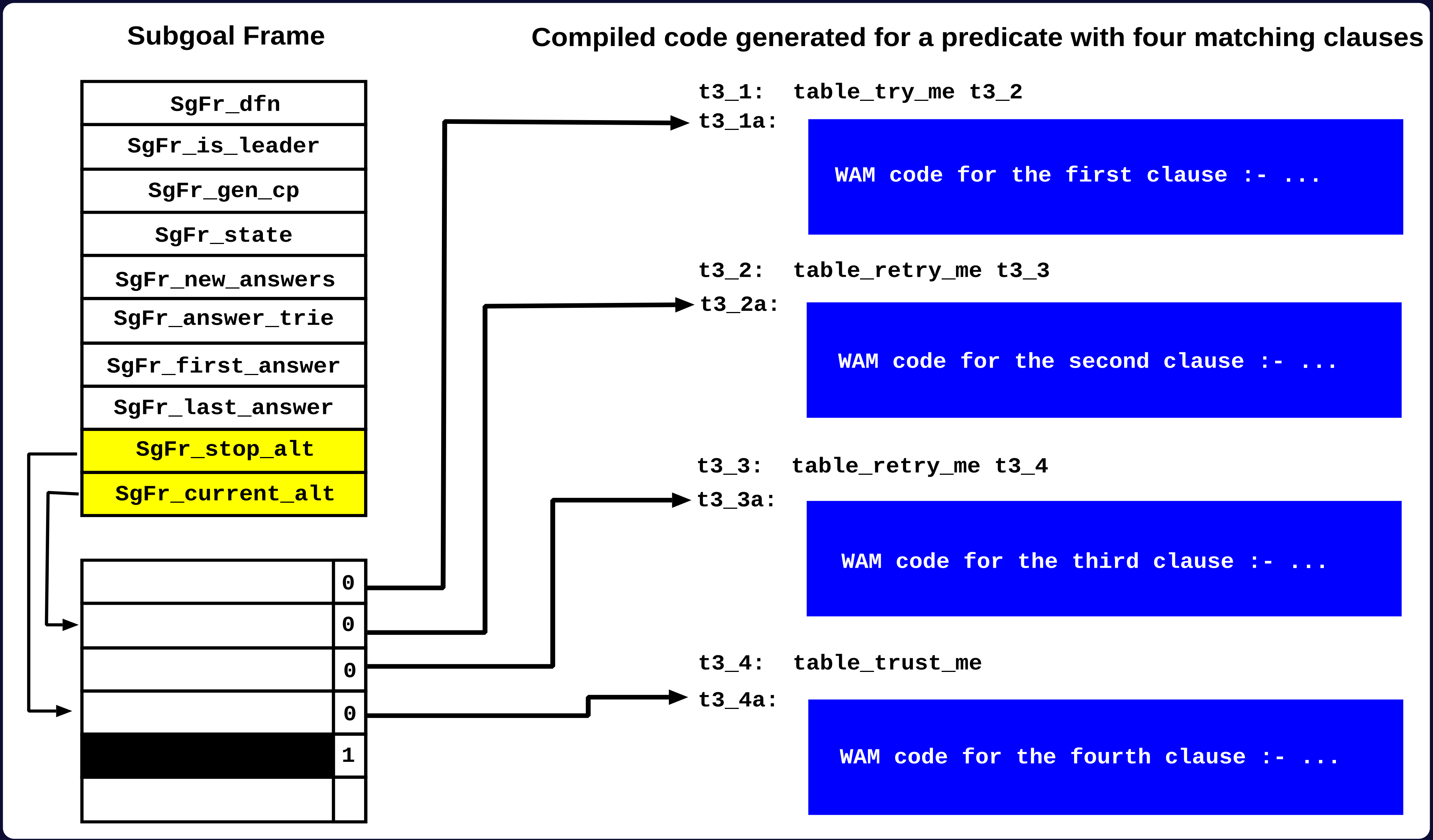
```
path_first(X,Z) :- ...
path_first(X,Z) :- ...

path_last(X,Z)  :- sld3 , edge(X,Z) , sld4.
path_last(X,Z)  :- sld1 , edge(X,Y) , path_last(Y,Z) , sld2.
```

Strategy	#SLD Computations			
	sld1/0	sld2/0	sld3/0	sld4/0
Recursive Clause Last				
Standard	48,602	352,277,129	48,602	205,920
DRE	1.00	1.00	1.00	1.00
DRA	1.00	**1.05**	**20.99**	**11.50**
DRS	1.00	**1.29**	1.00	1.00
DRE+DRA	1.00	1.05	20.99	11.50
DRE+DRS	1.00	1.29	1.00	1.00
DRA+DRS	1.00	**1.38**	20.99	11.50
All	1.00	1.38	20.99	11.50

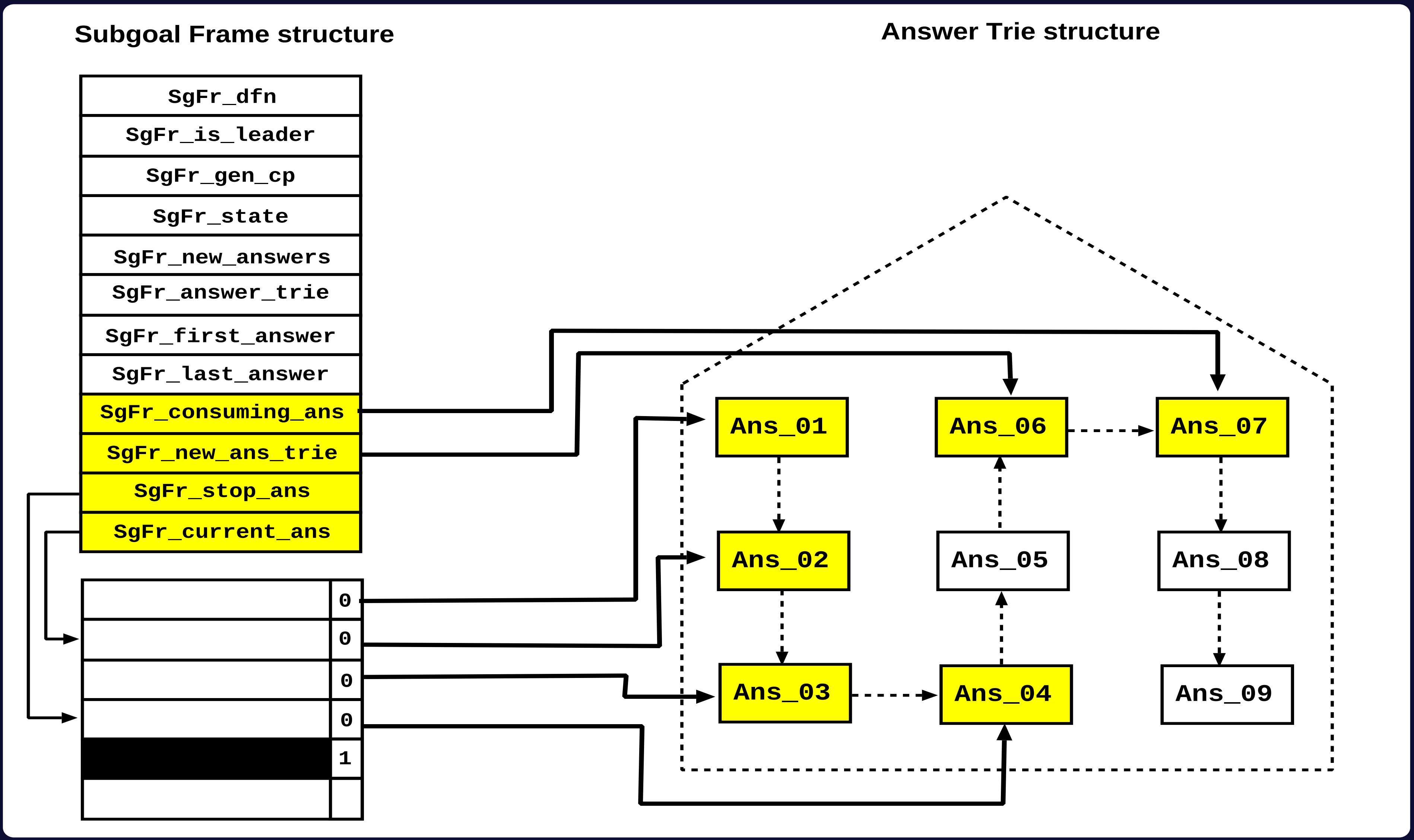
Conclusions and Further Work

- We have presented a new framework that integrates all possible combinations of the already existent linear tabling strategies **DRA** and **DRE** and the new strategy **DRS**.
- Our experiments for **DRS** strategy showed that the strategy of avoiding the consumption of non-looping solutions in re-evaluation rounds can be quite effective for programs that can benefit from it, with insignificant costs for the other programs.
- Further work will include study a possible source-code analysis tool that would determine which linear tabling strategy should be used for a particular program before it's evaluation.

Implementation Details - DRA



Implementation Details - DRS



Implementation Details - DRE