1 - Quiz

1. Which of the following statements about the unification algorithm are true?
   - ☐ The algorithm does not terminate if no *mgu* exists.
   - ☐ If the two input terms are not unifiable, an empty substitution is returned.
   - ☒ The occurs check avoids non-termination.
   - ☒ The algorithm always terminates.
   - ☒ Unification is used in Haskell's type inference algorithm.
2. Let ϕ be an *mgu* for the terms t_1 and t_2. Which of the following statements hold?
   - ☐ $\mathrm{ds}(\phi(t_1), t_2) = \{\}$
   - ☐ $t_1 = \phi(t_2)$
   - ☒ $\phi(t_1) = \phi(\phi(t_2))$
   - ☐ $t_1 \neq t_2$
   - ☒ $(\phi \circ \phi)(t_1) = \phi(t_2)$
3. Which of the following patterns unify with the expression `[42, true]`?
   - ☒ `[X | [true]]`
   - ☐ `[42 | true]`
   - ☒ `X`
   - ☒ `[42, False | X]`
   - ☐ `[[42] | X]`
4. Which of the following sentences are correct?
   - ☒ Prolog uses the selection strategy FIRST in the SLD resolution.
   - ☐ $\mathrm{ds}(f(a, g(Y), 73), f(X, b, 73)) = \{X, a, g(Y), b\}$
   - ☐ `?- g(X, Xs) = X.` has no solution in SWI-Prolog.
   - ☒ σ is a most general unifier, if for all unifiers σ' there exists a substitution ϕ such that $\sigma' = \phi \circ \sigma$.
   - ☐ `?- findall(X, member(X, [21, 42, 73]), L).` has more than one solution.
5. Which of the following queries are answered with `true` or a binding for the occuring variables.
   - ☐ `?- 42 + 31 is 31 + 42.`
   - ☒ `?- 42 is 7 * 6.`
   - ☐ `?- 20 + 1 = 22 - 1.`
   - ☒ `?- X = 42 + 31.`
   - ☐ `?- 21 * 2.`

1. Give all solutions of the goal `?- p(Y).`.

```
p(X) :- q(X), q(a), !.
p(X) :- q(X), !, q(c).
p(b).

q(b) :- q(a).
q(c).
```

The only solution is the following.

```
Y = c.
```

2. Give all solutions of the goal `?- append([1|Xs], [3|Ys], [1, 2, 3, 4, 1, 3]).`.

```
Xs = [2],
Ys = [4, 1, 3];
Xs = [2, 3, 4, 1],
Ys = [];
false.
```

3. Give the most general unifier (if it exists) for the terms `fun(g(Y, Z), [Y|[h(X)|[]]])` and `fun(X, [42|Xs])`. Otherwise, explain why no *mgu* exists.

$$\sigma = \{X \mapsto g(42, Z), Y \mapsto 42, Xs \mapsto [h(g(42, Z))|[]]\}$$

2 - Programming in Prolog

```
gt(s(_), o).
gt(s(N), s(M)) :- gt(N, M).

fromTo(N, N, [N]).
fromTo(N, M, [N|Xs]) :- gt(M, N), fromTo(s(N), M, Xs).
fromTo(N, M, [])     :- gt(N, M).

dropLess(_, [],     []).
dropLess(N, [N|Xs], [N|Xs]).
dropLess(N, [X|Xs], [X|Xs]) :- gt(X, N).
dropLess(N, [X|Xs], Ys)     :- gt(N, X), dropLess(N, Xs, Ys).
```

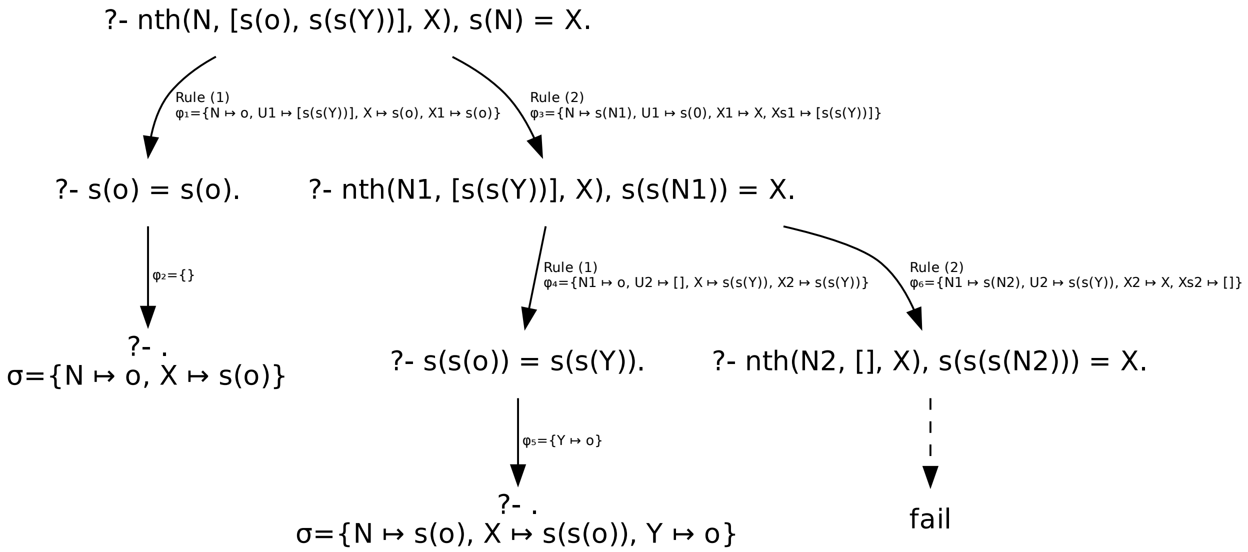

Abbildung 1: SLD-Resolution-Tree for the goal '?- nth(N, [s(o), s(s(Y))], X), s(N) = X.'