Das Lösen einer Anfrage mit Backtracking

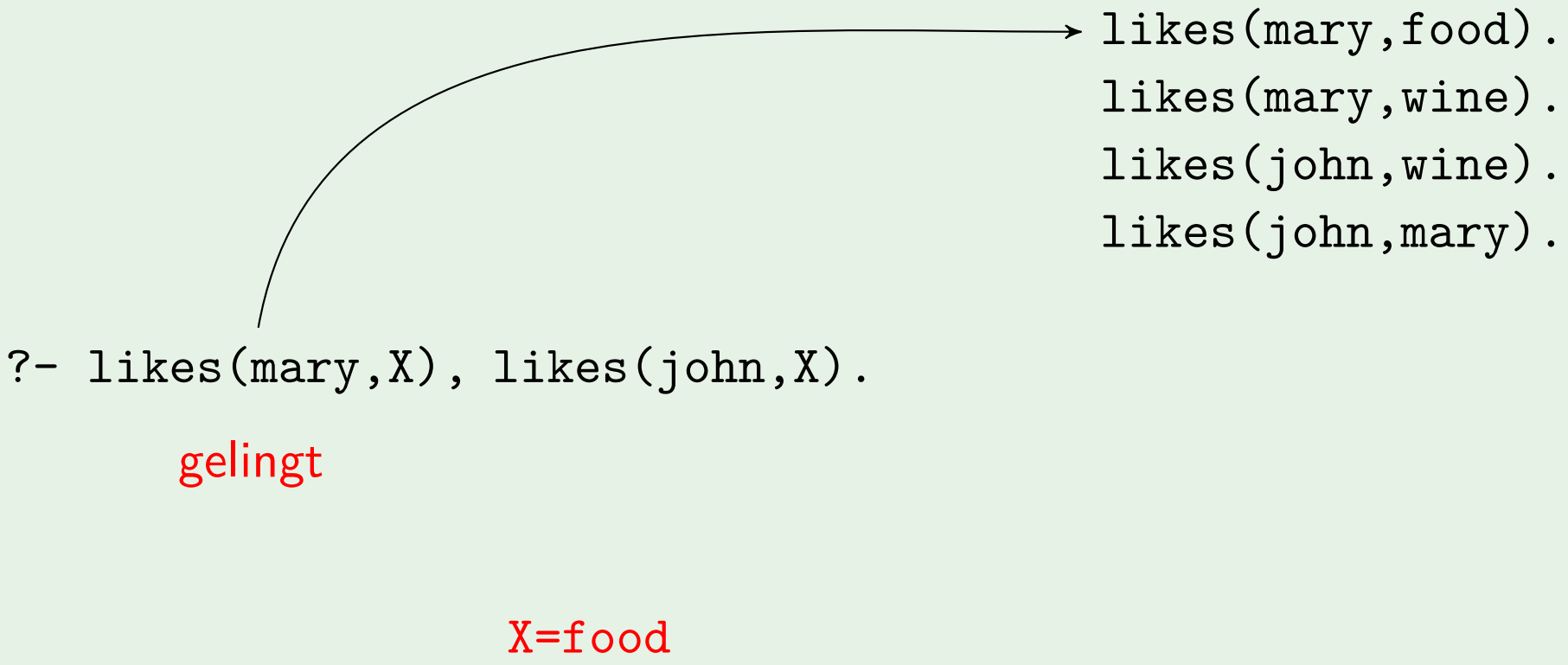
Beispiel aus Clocksin Mellish S.14/15

```
likes(mary,food).
likes(mary,wine).
likes(john,wine).
likes(john,mary).
```

```
?- likes(mary,X), likes(john,X).
```

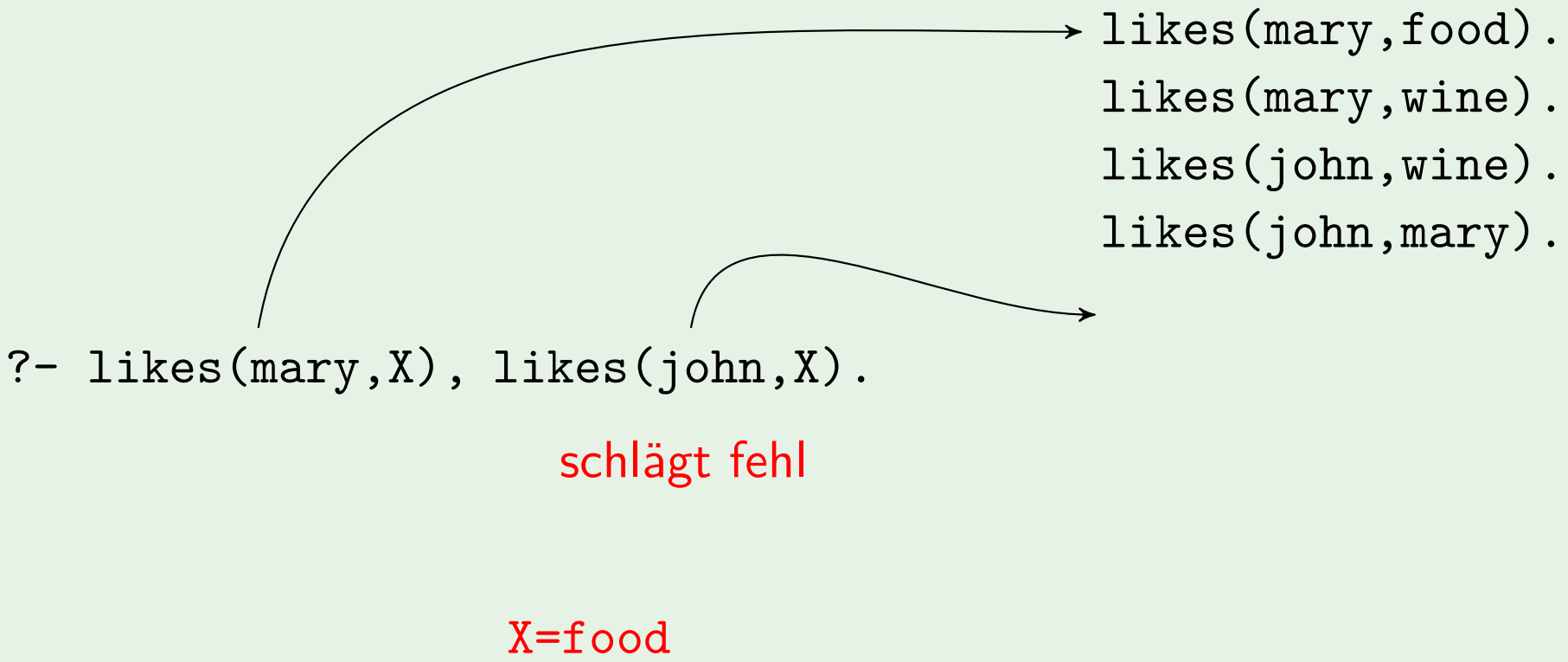
Das Lösen einer Anfrage mit Backtracking

Beispiel aus Clocksin Mellish S.14/15



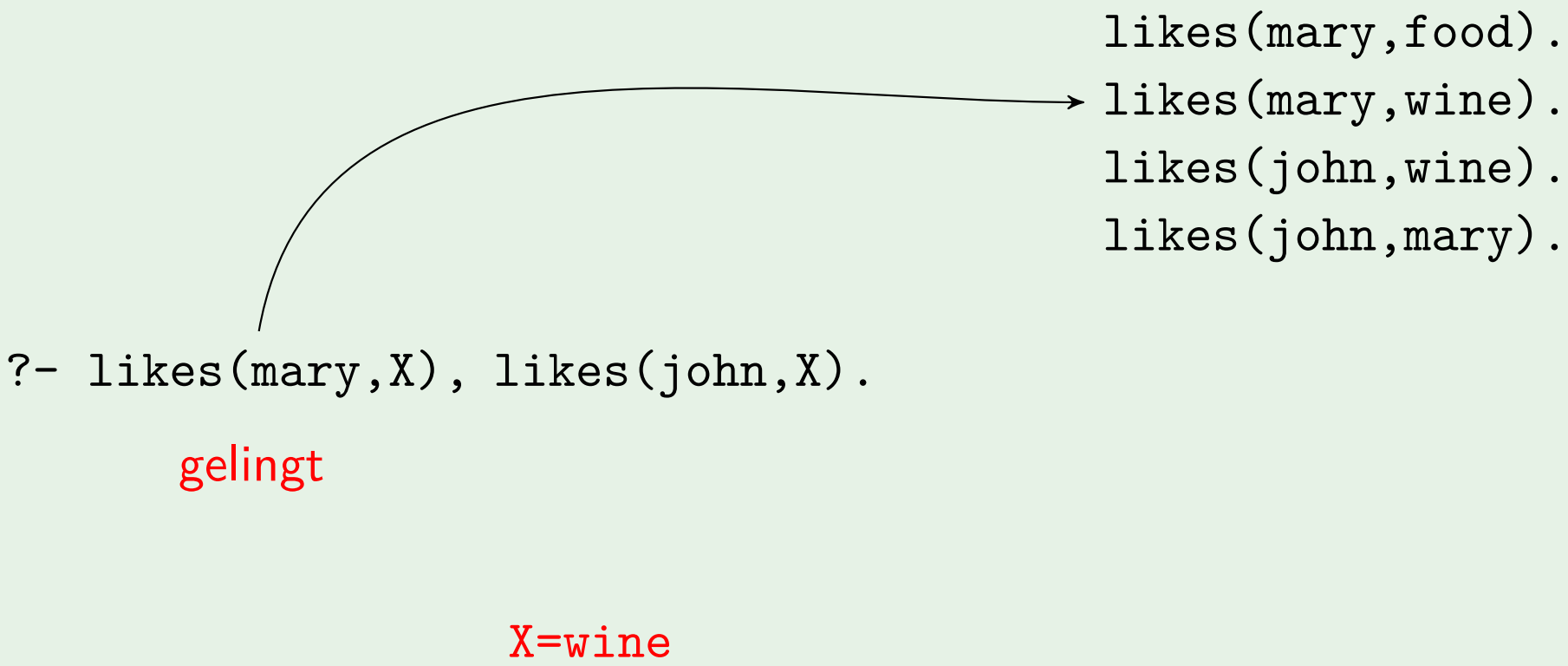
Das Lösen einer Anfrage mit Backtracking

Beispiel aus Clocksin Mellish S.14/15



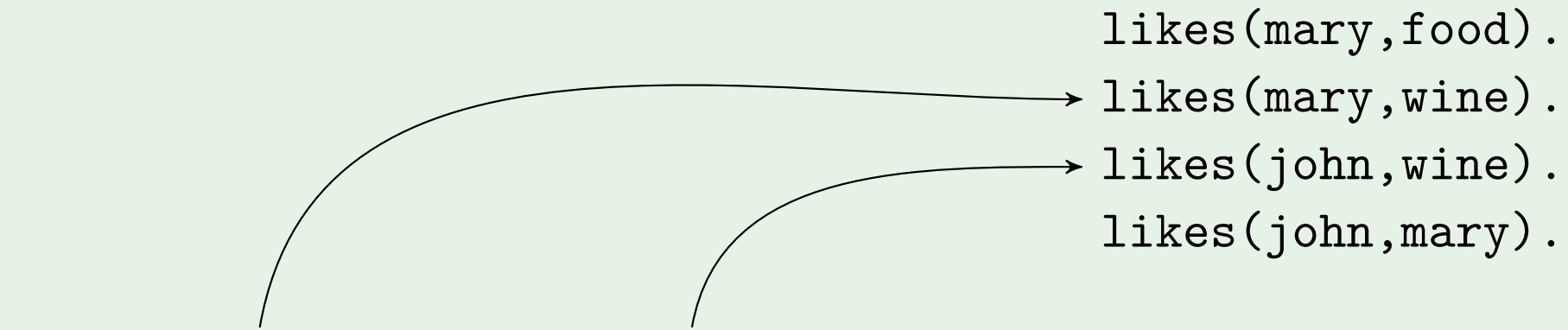
Das Lösen einer Anfrage mit Backtracking

Beispiel aus Clocksin Mellish S.14/15



Das Lösen einer Anfrage mit Backtracking

Beispiel aus Clocksin Mellish S.14/15

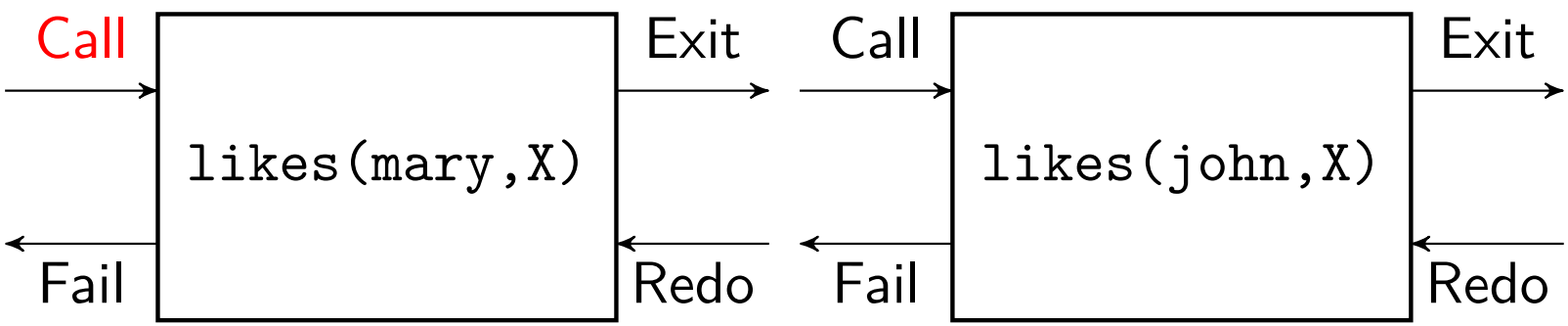


gelingt

X=wine

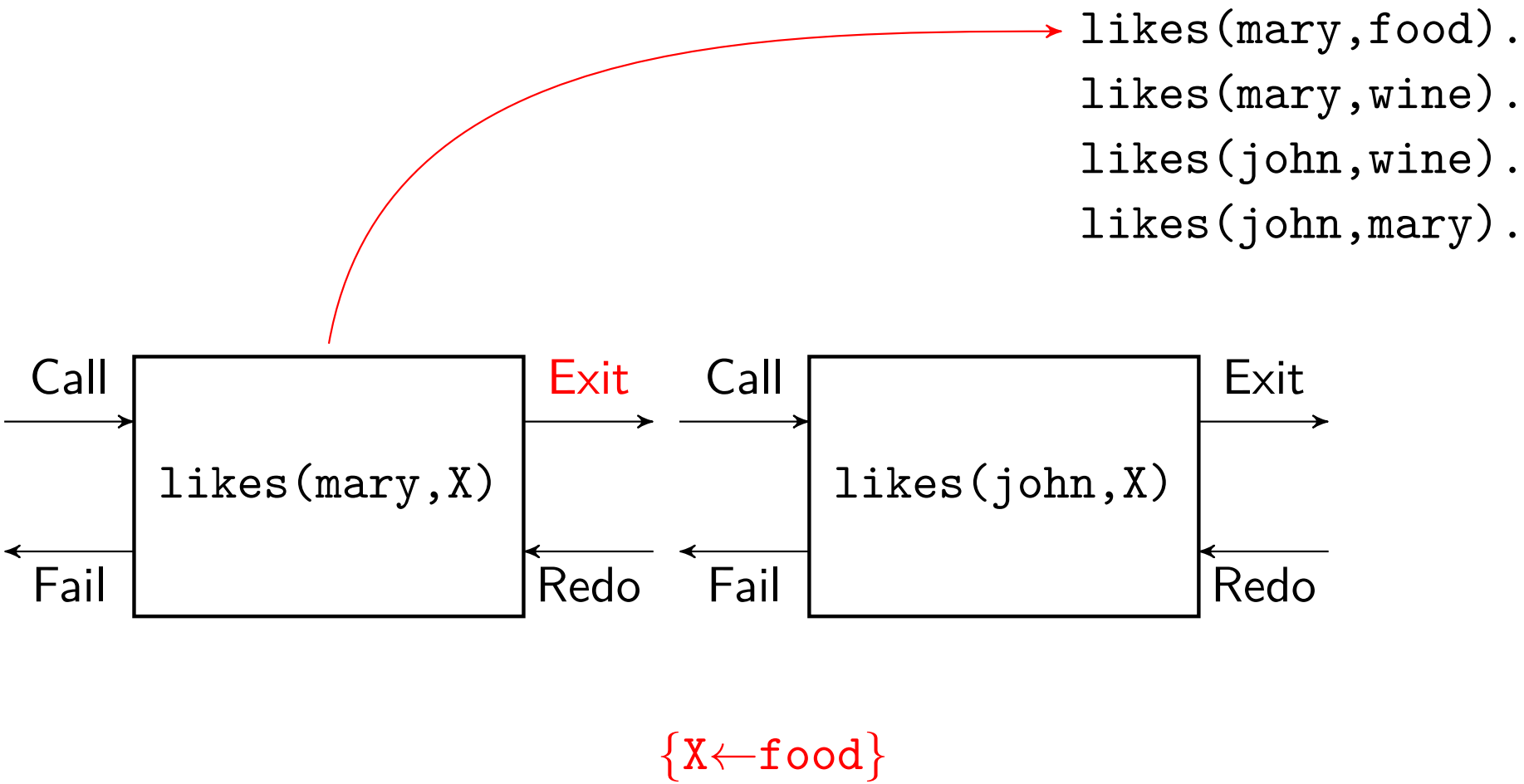
Byrd Box Model

```
likes(mary,food).
likes(mary,wine).
likes(john,wine).
likes(john,mary).
```

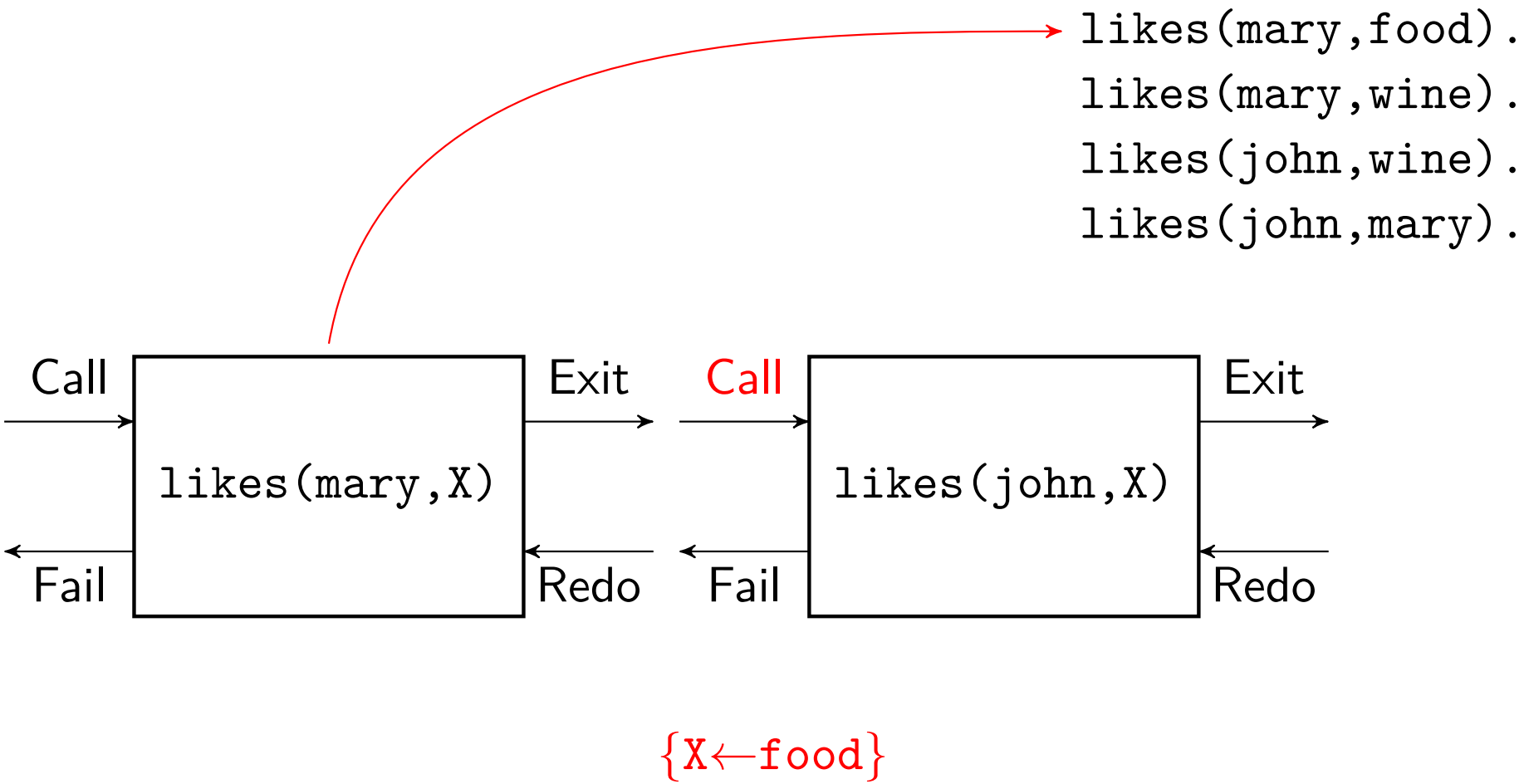


{}

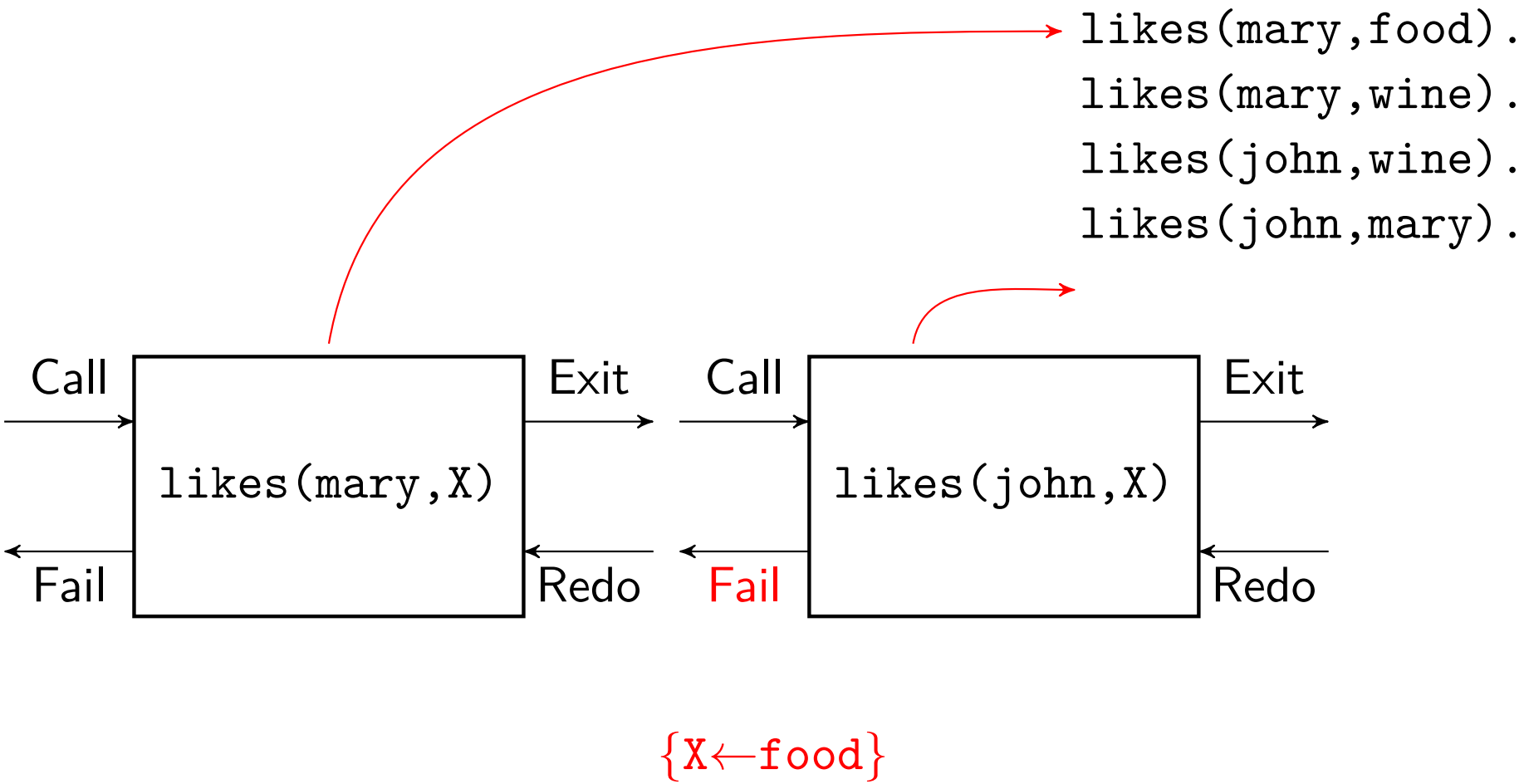
Byrd Box Model



Byrd Box Model

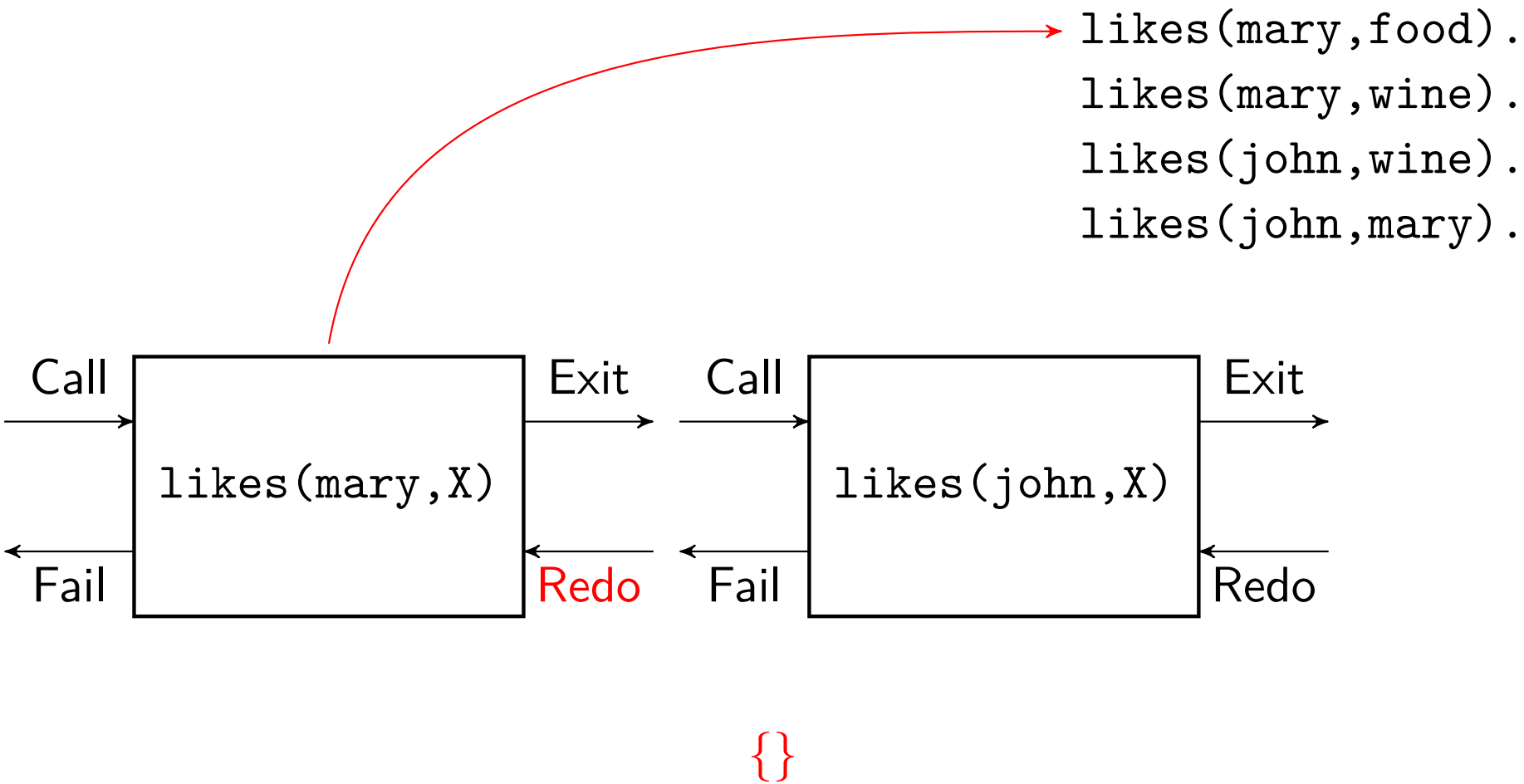


Byrd Box Model

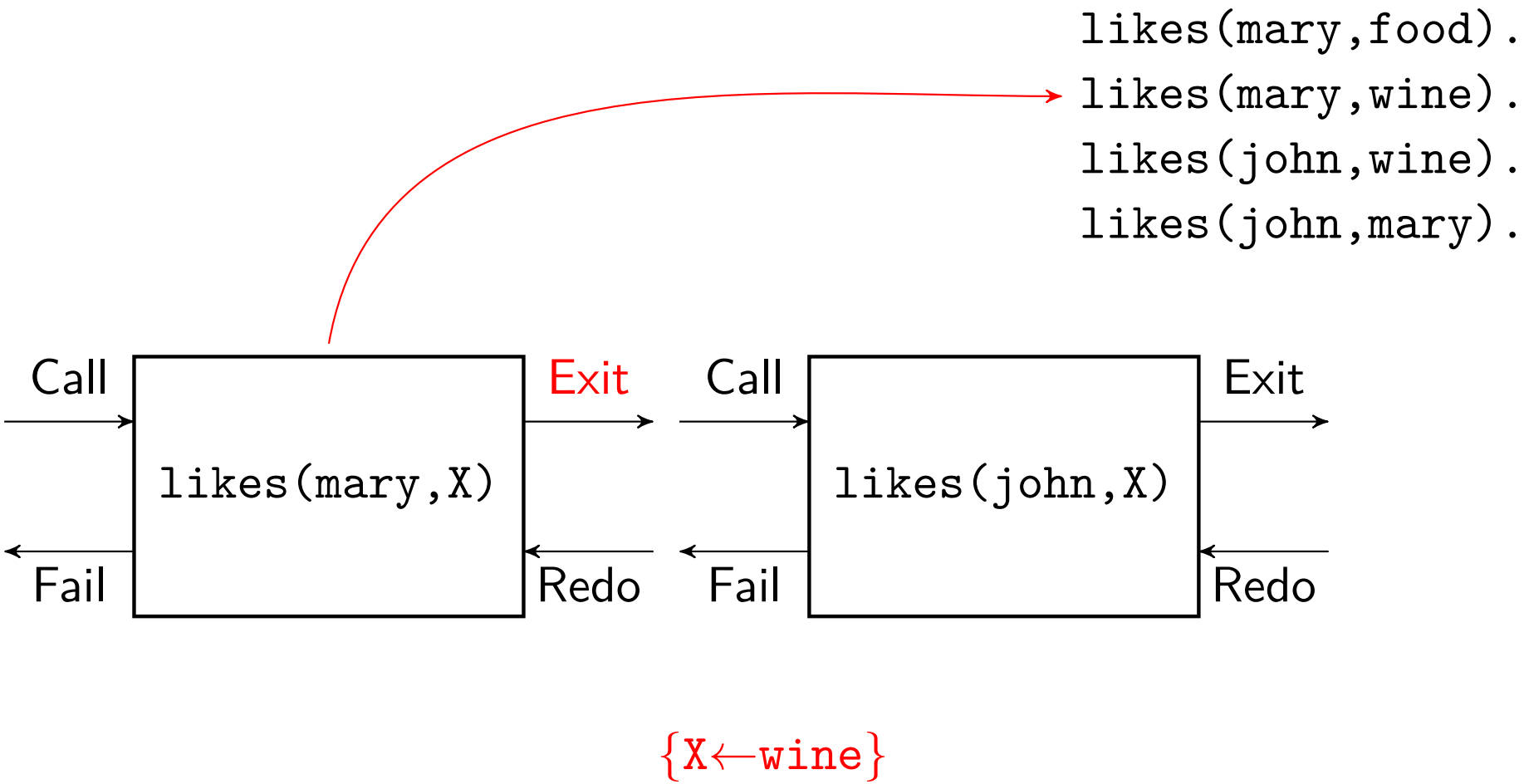


{X←food}

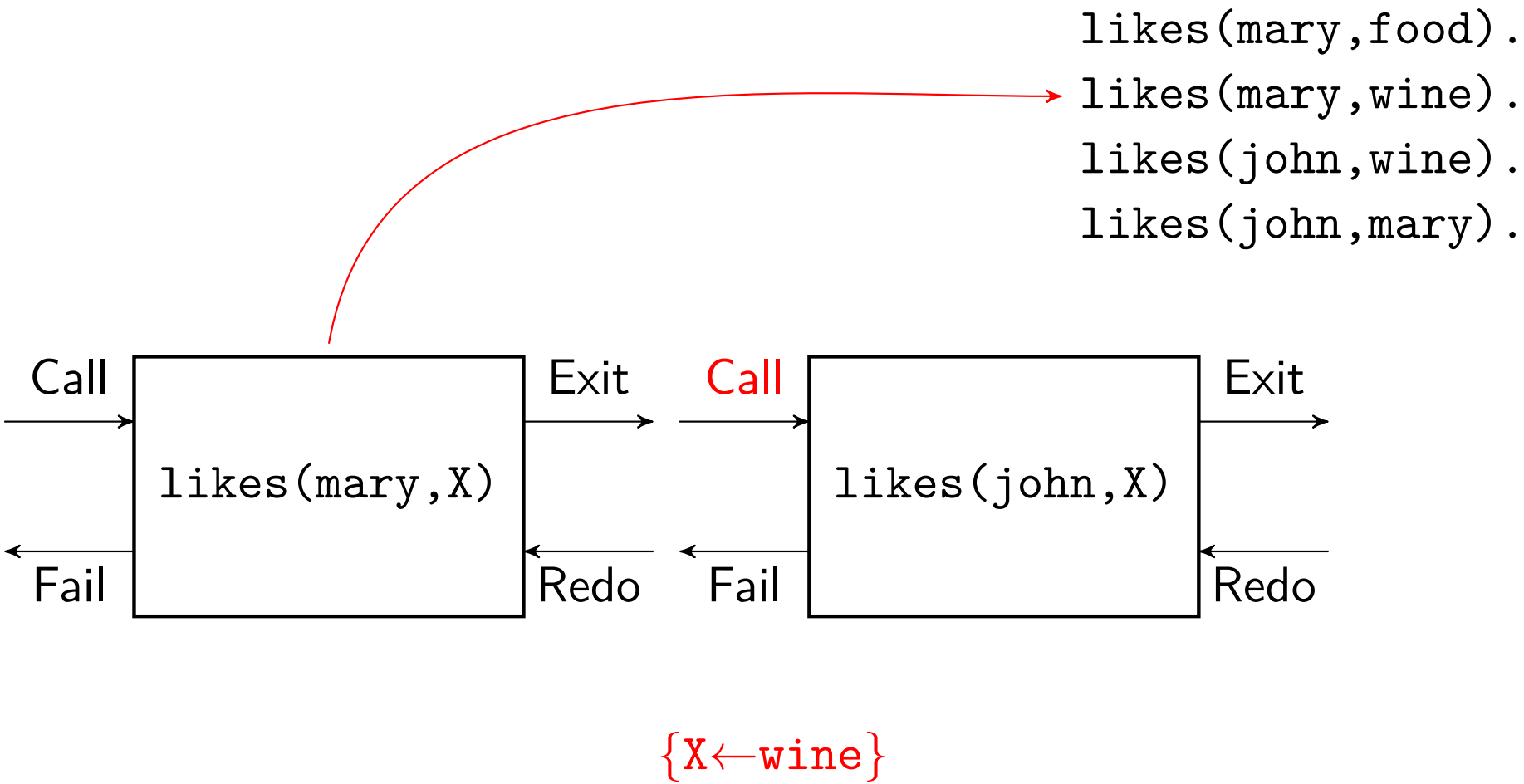
Byrd Box Model



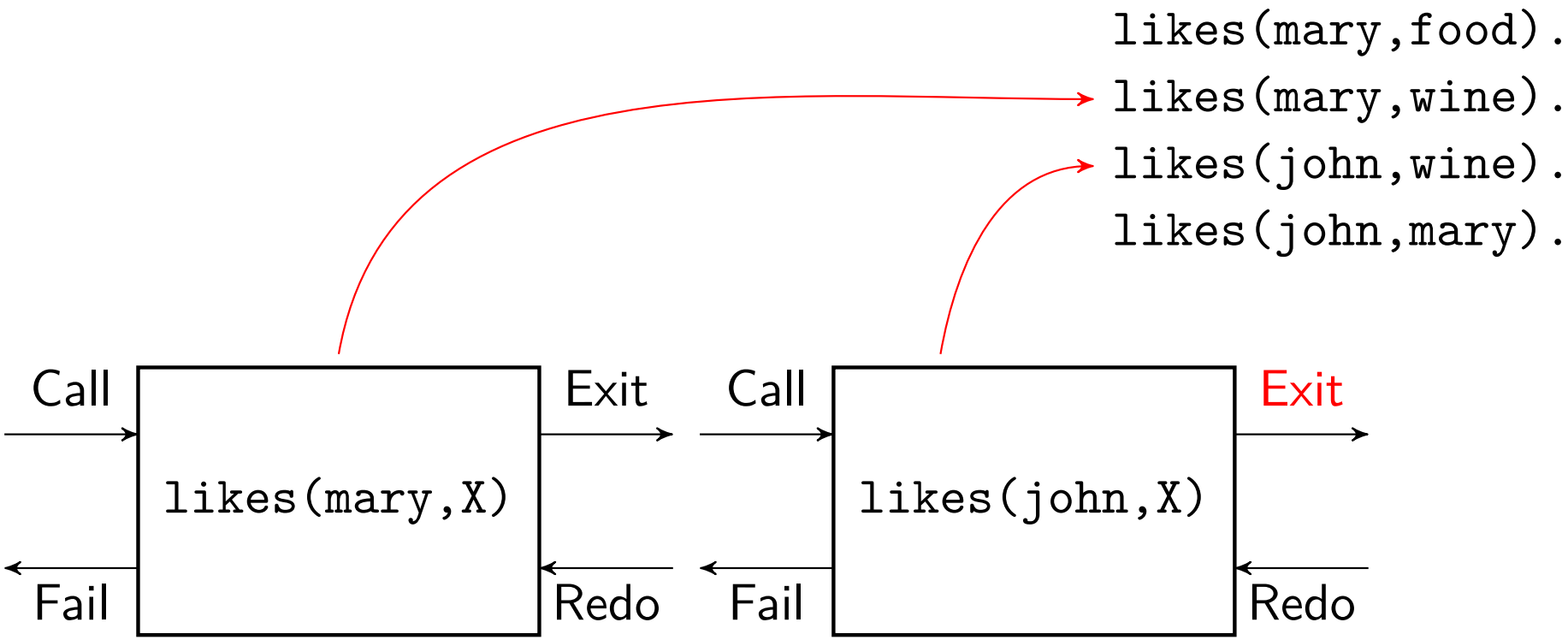
Byrd Box Model



Byrd Box Model



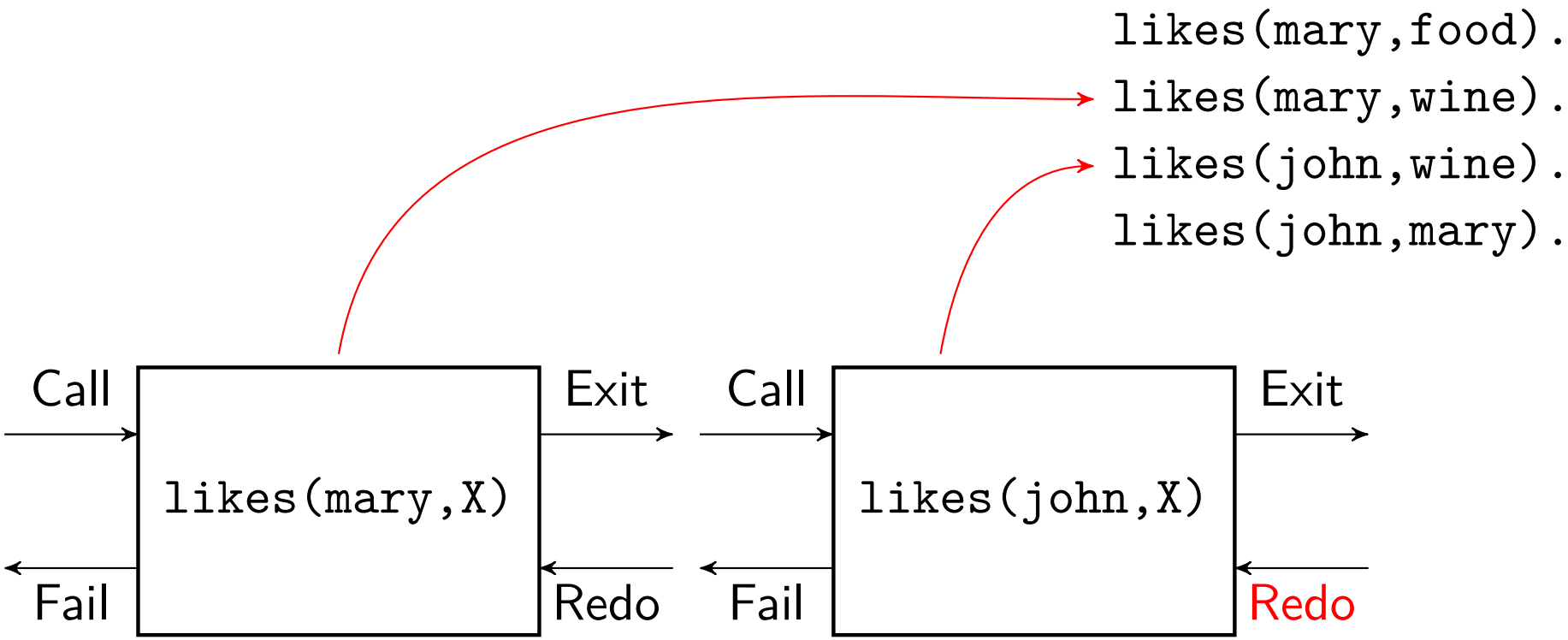
Byrd Box Model



{X←wine}

Prolog antwortet: X=wine Benutzer gibt ; ein ...

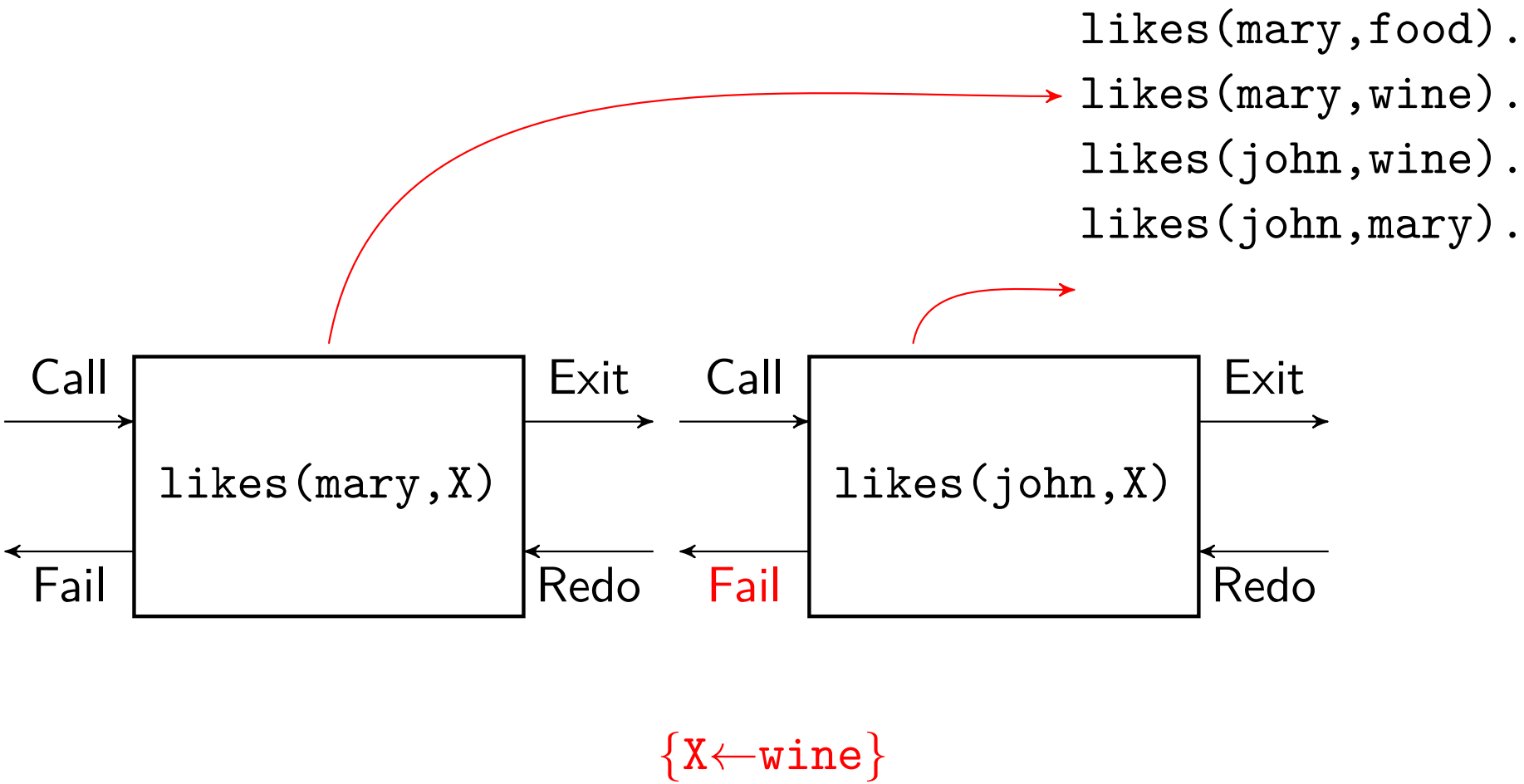
Byrd Box Model



{X←wine}

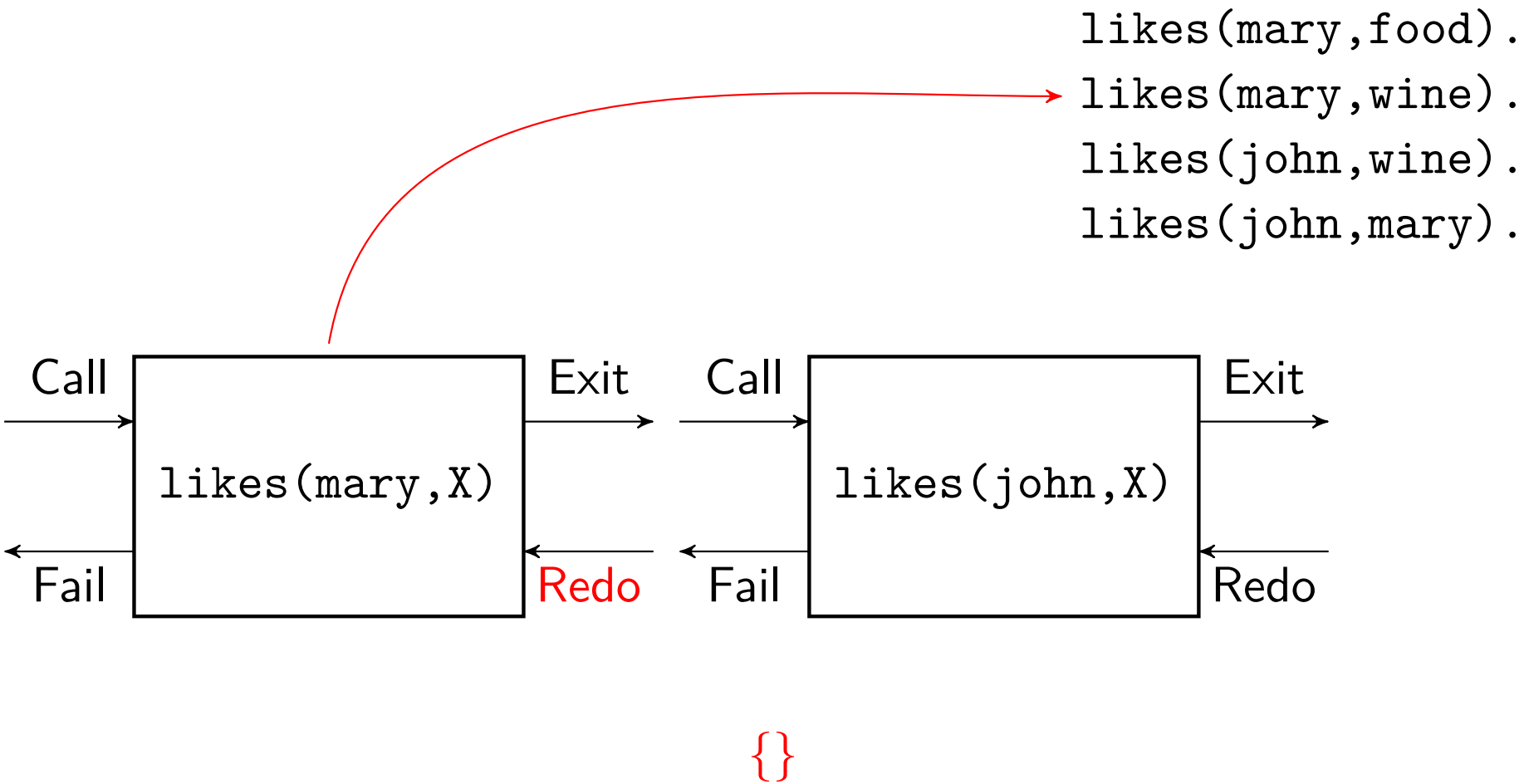
. . . und erzwingt damit Backtracking (redo).

Byrd Box Model



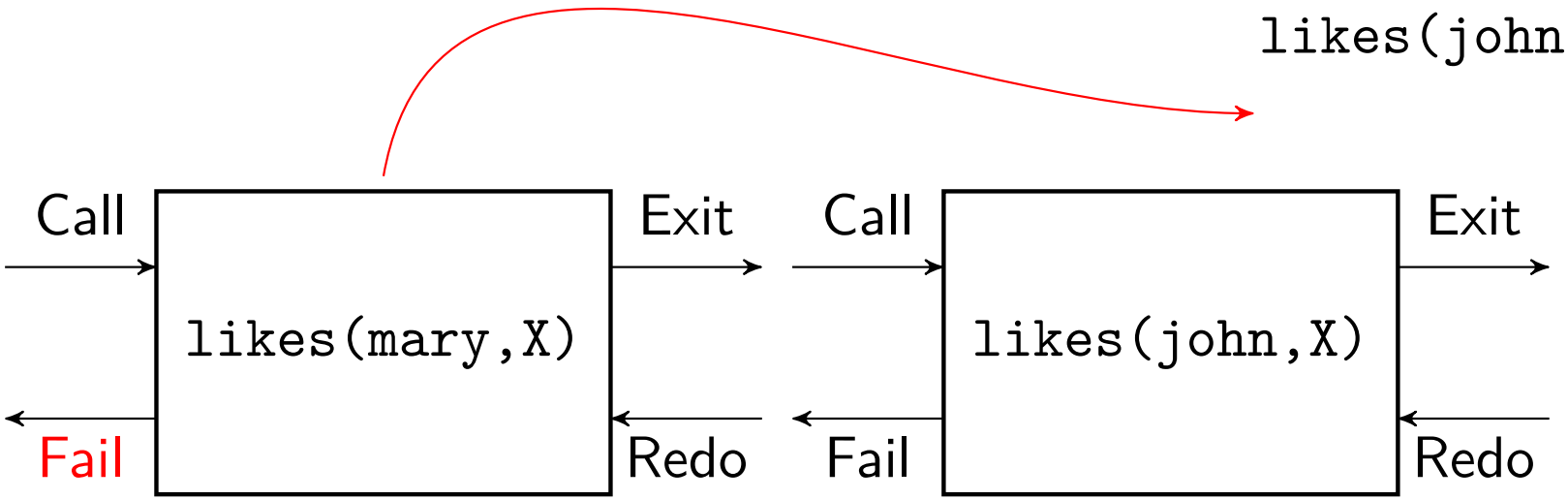
{X←wine}

Byrd Box Model



Byrd Box Model

```
likes(mary,food).
likes(mary,wine).
likes(john,wine).
likes(john,mary).
```

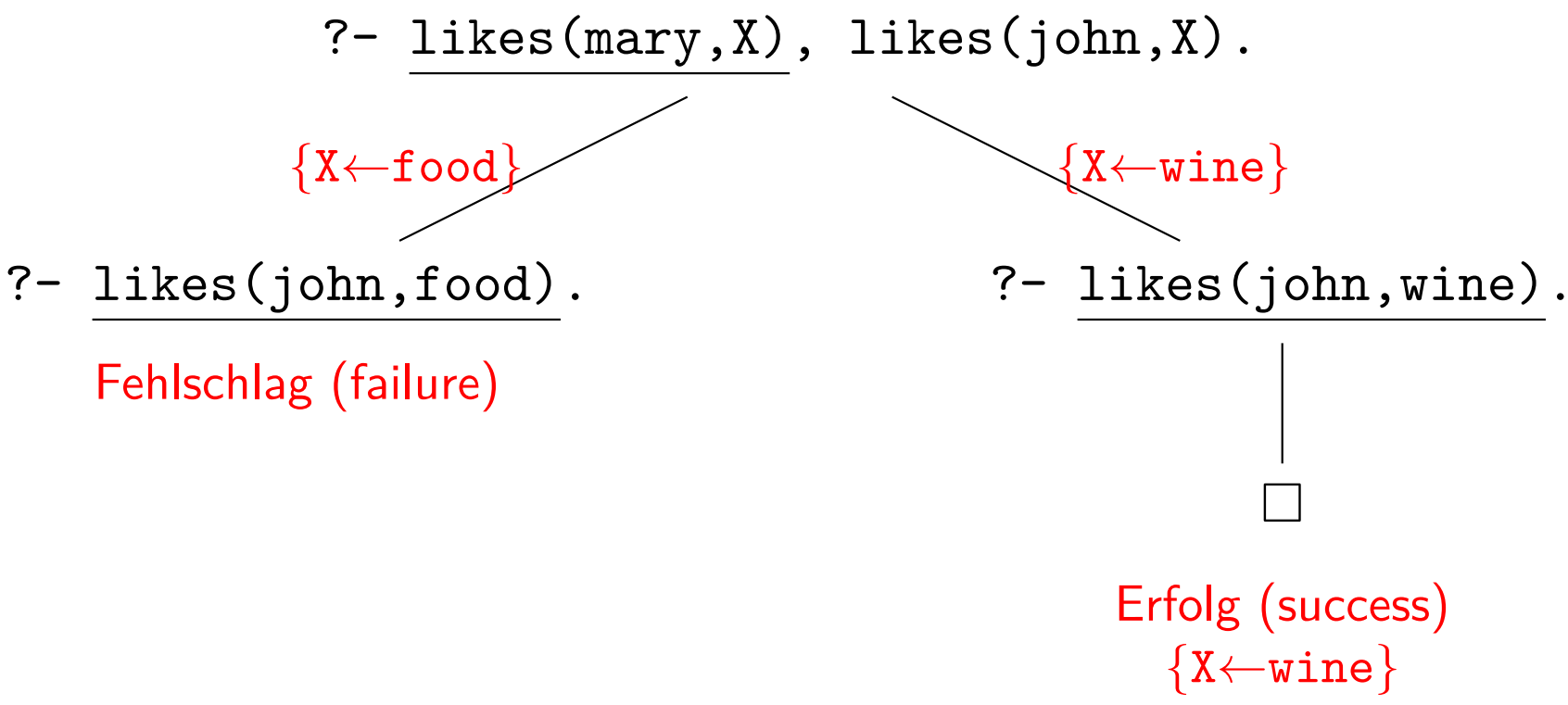


{}

Anfrage schlägt fehl.

SLD-Baum

```
likes(mary,food).
likes(mary,wine).
likes(john,wine).
likes(john,mary).
```



Jeder Erfolgsast liefert eine Antwortsubstitution, hier {X←wine}.

Prolog antwortet: X=wine.

Unifikation

- unifizieren = zu einem machen
- Gleichungslösen

Beispiel

- `?- f(X,g(X),a) = f(h(Y),Z,Y).`
  `X = h(a), Y = a, Z = g(h(a)).`
- Die Substitution {`X`←`h(a)`, `Y`←`a`, `Z`←`g(h(a))`}
  - unifiziert beide Seiten der Gleichung.
  - heißt Unifikator dieser beiden Terme.

Substitutionen und Unifikation

Substitution allgemein

- $\{X_1 \leftarrow t_1, \ldots, X_n \leftarrow t_n\}$
  Dabei sind $X_1, \ldots, X_n$ Variable und $t_1, \ldots, t_n$ Terme.
- Anwendung der Substitution auf einen Term t:
  $t\{X_1 \leftarrow t_1, \ldots, X_n \leftarrow t_n\}$ ist der Term, der sich aus dem Term t ergibt, wenn man darin jedes X_1 durch t_1 ersetzt und ... und jedes X_n durch t_n ersetzt.
- Wir bezeichnen Substitutionen mit kleinen griechischen Buchstaben σ, τ, ...

Beispiel

- Substitution $\{$`X`$\leftarrow$`h(a)`, `Y`$\leftarrow$`a`, `Z`$\leftarrow$`g(h(a))`$\}$
- Term `f(X,g(X),a)`
- `f(X,g(X),a)` $\{$`X`$\leftarrow$`h(a)`, `Y`$\leftarrow$`a`, `Z`$\leftarrow$`g(h(a))`$\}$ ist der Term `f(h(a)),g(h(a)),a)`
- `f(h(Y),Z,Y)` $\{$`X`$\leftarrow$`h(a)`, `Y`$\leftarrow$`a`, `Z`$\leftarrow$`g(h(a))`$\}$ ist ebenfalls der Term `f(h(a)),g(h(a)),a)`

Definition

Ein Unifikator zweier Terme s und t ist eine Substitution σ mit $s\sigma = t\sigma$. ($s\sigma$ formal gleich $t\sigma$)

Beispiel

Die Substitution $\{$`X`$\leftarrow$`h(a)`, `Y`$\leftarrow$`a`, `Z`$\leftarrow$`g(h(a))`$\}$ ist ein Unifikator der Terme `f(X,g(X),a)` und `f(h(Y),Z,Y)`.

Komposition von Substitutionen

Definition (Komposition)

Die Komposition $\sigma\tau$ zweier Sustitutionen σ und τ ist definiert durch

$$X(\sigma\tau) = (X\sigma)\tau \qquad \text{für alle Variablen } X.$$

Beispiel

Sei $\sigma = \{X \leftarrow f(X, Y)\}$ und $\tau = \{Y \leftarrow g(X)\}$. Dann ist

- $X(\sigma\tau) = (X\sigma)\tau = f(X, Y)\tau = f(X, g(X))$
- $Y(\sigma\tau) = (Y\sigma)\tau = Y\tau = g(X)$
- $Z(\sigma\tau) = (Z\sigma)\tau = Z\tau = Z$ für alle anderen Variablen Z,

also $\sigma\tau = \{X \leftarrow f(X, g(X)), Y \leftarrow g(X)\}$.

Für jeden Term t gilt $t(\sigma\tau) = (t\sigma)\tau$.
Wir dürfen die Klammern weglassen: $t\sigma\tau$.

Nichtübereinstimmungsmenge zweier Terme

Definition

- Seien s und t zwei verschiedene Terme.
- Betrachte die erste Stelle, an der sich s und t unterscheiden.
- Die beiden Teilterme von s bzw. t, die an dieser Stelle beginnen, bilden die Nichtübereinstimmungsmenge (disagreement set) der Terme s und t.

Beispiel

Die Nichtübereinstimmungsmenge der beiden Terme

- `f(X,g(X),a)`
- `f(h(Y),Z,Y)`

ist `{X,h(Y)}` (oben rot gekennzeichnet).

Unifikationsalgorithmus

Definition

Ein Unifikationsalgorithmus ist ein Algorithmus zum Finden eines Unifikators zweier Terme s und t.

Idee des Unifikationsalgorithmus

Konstruiere nacheinander Terme s_i und t_i und Substitutionen σ_i:

- $s_0 := s$ und $t_0 := t$.
- Wenn s_i und t_i verschieden sind, dann sind s und t nur dann unifizierbar, wenn die Nichtübereinstimmungsmenge die Form $\{X_i, r_i\}$ mit einer Variablen X_i und einem Term r_i hat, der X_i nicht enthält.
- Dann sei $\sigma_{i+1} := \{X_i \leftarrow r_i\}$, $\quad s_{i+1} := s_i\sigma_{i+1}$ und $t_{i+1} := t_i\sigma_{i+1}$.
- Wenn $s_n = t_n$ ist, dann ist $\sigma_1 \ldots \sigma_n$ der gesuchte Unifikator.

Beispiel zum Unifikationsalgorithmus

f(X,g(X),a)
f(h(Y),Z,Y)

$$\sigma_1 = \{\mathtt{X} \leftarrow \mathtt{h(Y)}\}$$

f(h(Y),g(h(Y)),a)
f(h(Y),Z,Y)

$$\sigma_2 = \{\mathtt{Z} \leftarrow \mathtt{g(h(Y))}\}$$

f(h(Y),g(h(Y)),a)
f(h(Y),g(h(Y)),Y)

$$\sigma_3 = \{\mathtt{Y} \leftarrow \mathtt{a}\}$$

f(h(a),g(h(a)),a)
f(h(a),g(h(a)),a)

Unifikator $\sigma_1\sigma_2\sigma_3 = \{\mathtt{X} \leftarrow \mathtt{h(a)},\ \mathtt{Z} \leftarrow \mathtt{g(h(a))},\ \mathtt{Y} \leftarrow \mathtt{a}\}$

Clash und Cycle

- Wenn eine der Nichtübereinstimmungsmengen aus zwei Termen $f(u_1, \ldots, u_m)$ und $g(v_1, \ldots, v_n)$ besteht mit verschiedenen Funktoren f und g, dann sind s und t nicht unifizierbar. Wir sprechen dann von einem Clash.
- Wenn eine der Nichtübereinstimmungsmengen aus einer Variablen und einem Term, in dem diese Variable vorkommt, besteht, dann sind s und t nicht unifizierbar. Wir sprechen dann von einem Cycle.

Die Überprüfung, ob eine Variable in einem Term vorkommt (englisch occurs) heißt Occurs Check.

Beispiel zum Unifikationsalgorithmus

`f(X)`
`g(a)`

Clash — nicht unifizierbar, weil $\mathtt{f}/1 \neq \mathtt{g}/1$.

Beispiel zum Unifikationsalgorithmus

`f(X,g(a,Y))`
`f(Y,h(Y,Z))`

$$\sigma_1 = \{\mathtt{Y} \leftarrow \mathtt{X}\}$$

`f(X,g(a,X))`
`f(X,h(X,Z))`

Clash — nicht unifizierbar, weil $\mathtt{g}/2 \neq \mathtt{h}/2$.

In Prolog sind gleichnamige verschiedenstellige Funktoren f/m und f/n erlaubt, gelten aber als verschieden.

Beispiel zum Unifikationsalgorithmus

```
f(X,Y)
f(a,b,c)
```

Clash — in Prolog nicht unifizierbar, weil $\texttt{f}/2 \neq \texttt{f}/3$.

Beispiel zum Unifikationsalgorithmus

`X`
`f(X)`

Cycle — nicht unifizierbar, weil `X` in `f(X)` vorkommt.

Beispiel zum Unifikationsalgorithmus

`f(X)`
`f(g(X))`

Cycle — nicht unifizierbar, weil `X` in `g(X)` vorkommt.

Ein Cycle wird entdeckt durch einen Occurs Check.

Allgemeinere Substitution

Definition

Eine Substitution σ heißt allgemeiner als eine Substitution τ, wenn es eine Substitution ρ gibt, sodass $\tau = \sigma\rho$.

Beispiel

- $\sigma := \{X \leftarrow f(Y)\}$ ist eine Unifikator der Terme X und $f(Y)$.
- Auch $\tau := \{X \leftarrow f(a),\ Y \leftarrow a\}$ ist ein Unifikator von X und $f(Y)$.
- σ ist allgemeiner als τ, weil $\tau = \sigma\rho$ für $\rho := \{Y \leftarrow a\}$.

Satz

Sei σ allgemeiner als τ. Dann gilt:
Wenn σ Unifikator von s und t ist, dann auch τ.

- Je allgemeiner σ ist, desto stärker ist die Aussage „σ ist Unifikator von s und t“.
- Wir wollen einen möglichst allgemeinen Unifikator von s und t haben.

Allgemeinster Unifikator

Definition

Ein Unifikator σ zweier Terme s und t heißt allgemeinster Unifikator (most general unifier, mgu) von s und t, wenn σ allgemeiner als jeder Unifikator von s und t ist.

Satz

Seien s und t zwei Terme.

- Wenn s und t unifizierbar sind, dann liefert der Unifikationsalgorithmus für s und t einen mgu.
- Wenn s und t nicht unifizierbar sind, dann meldet der Unifikationsalgorithmus für s und t, dass sie nicht unifizierbar sind (Clash oder Cycle).

Occurs Check in Prolog

Der Occurs Check

- ist zeitaufwändig.
- wird bei geeigneter Programmiertechnik meist nicht benötigt.
- wird daher in Prolog standardmäßig weggelassen.

Beispiel

- `?- X=f(X).`
  Reaktion von Prolog undefiniert, wenn nicht der Occurs Check explizit eingeschaltet worden ist (hängt vom jeweiligen Prologsystem ab).
- `?- unify_with_occurs_check(X,f(X)).`
  schlägt fehl.