Unifitseerimine

- Muutuja X esinemisi tähistame $\tilde{X}$.
- Transleeritakse erinevalt sõltuvalt sellest, kas on initsialiseeritud või mitte.
- Kasutame makrot put $\tilde{X}$ ρ:

$$
\begin{aligned}
\text{put } X\ \rho &= \texttt{putvar}\ (\rho\ X) \\
\text{put } \bar{X}\ \rho &= \texttt{putref}\ (\rho\ X) \\
\text{put } \_\ \rho &= \texttt{putanon}
\end{aligned}
$$

Unifitseerimine

Unifitseerimise $\tilde{X} = t$ transleerimine:

- lisame magasini viida muutujale X;
- konstrueerime kuhja termi t esituse;
- toome sisse uue käsu, mis reliseerib unifitseermise.

$$\begin{aligned} \text{code}_G\ (\tilde{X} = t)\ \rho \quad = \quad & \text{put}\ \tilde{X}\ \rho \\ & \text{code}_A\ t\ \rho \\ & \texttt{unify} \end{aligned}$$

Unifitseerimine

Näide: olgu antud võrrand

$$\bar{U} = f(g(\bar{X}, Y), a, Z)$$

Siis keskkonna

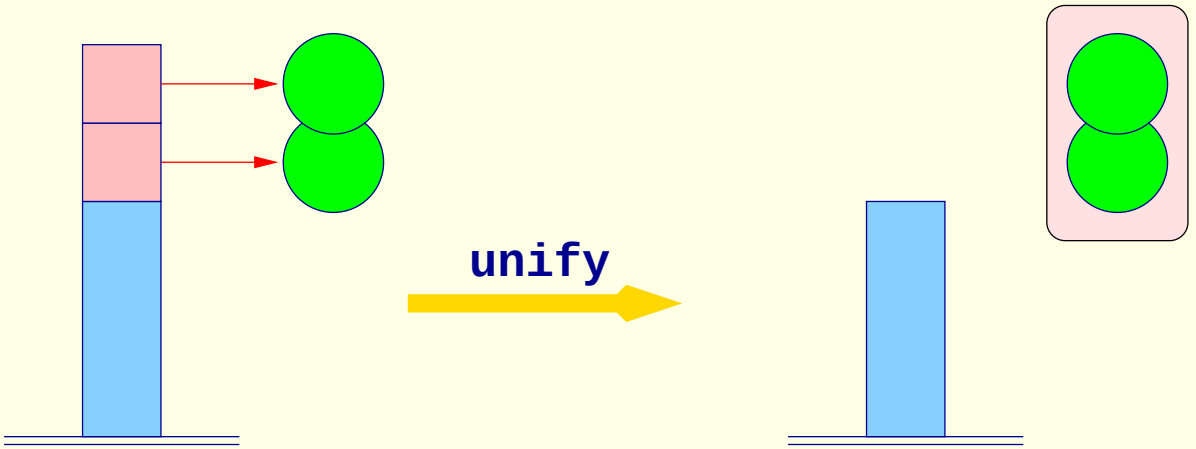
$$\rho = \{X \mapsto 1,\ Y \mapsto 2,\ Z \mapsto 3,\ U \mapsto 4\}$$

korral emiteeritakse kood:

```
putref 4          putatom a
putref 1          putvar 3
putvar 2          putstruct f/3
putstruct g/2     unify
```

Unifitseerimine

Käsk `unify` rakendab täitmisaegset funktsiooni `unify()` kahele ülemisele viidale:




```
unify (S[SP−1], S[SP−2]);
SP = SP − 2;
```

Unifitseerimine

Funktsioon `unify()`

- . . . võtab kaks kuhja aadressit. Iga väljakutse jaoks garanteerime, et viidaahelad oleksid elimineeritud.
- . . . kontrollib, kas need aadressid on juba *identsed*. Sel juhul ei tee midagi ja unifitseerimine õnnestus.
- . . . seob *nooremad muutujad* (suuremad aadressid) *vanemate muutujatega* (väiksemate aadressitega).
- . . . muutuja sidumisel termiga kontrollib, kas muutuja esineb termis või mitte (occur-check).
- . . . salvestab loodud seosed.
- . . . võib ebaõnnestuda (fail), misjuhul toimub *tagasipöördumine* (backtracking).

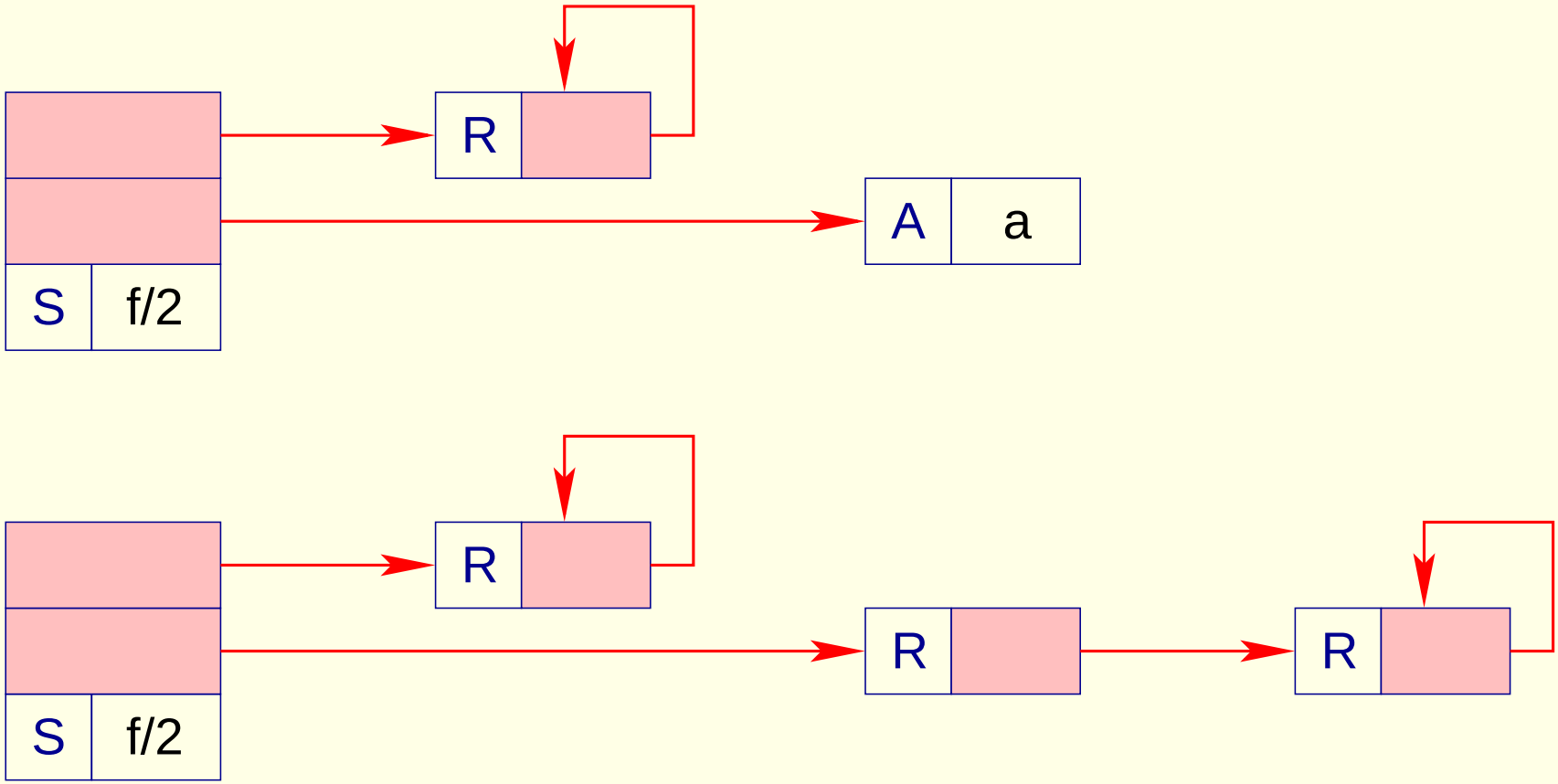
Unifitseerimine

```
bool unify (ref u, ref v) {
  if (u == v) return true;
  if (H[u] == (R,_)) {
    if (H[v] == (R,_)) {
      if (u > v) {
        H[u] = (R,v); trail(u); return true;
      } else {
        H[v] = (R,u); trail(v); return true;
      }
    } else if (check (u,v)) {
      H[u] = (R,v); trail(u); return true;
    } else { backtrack(); return false; }
  }
...
```

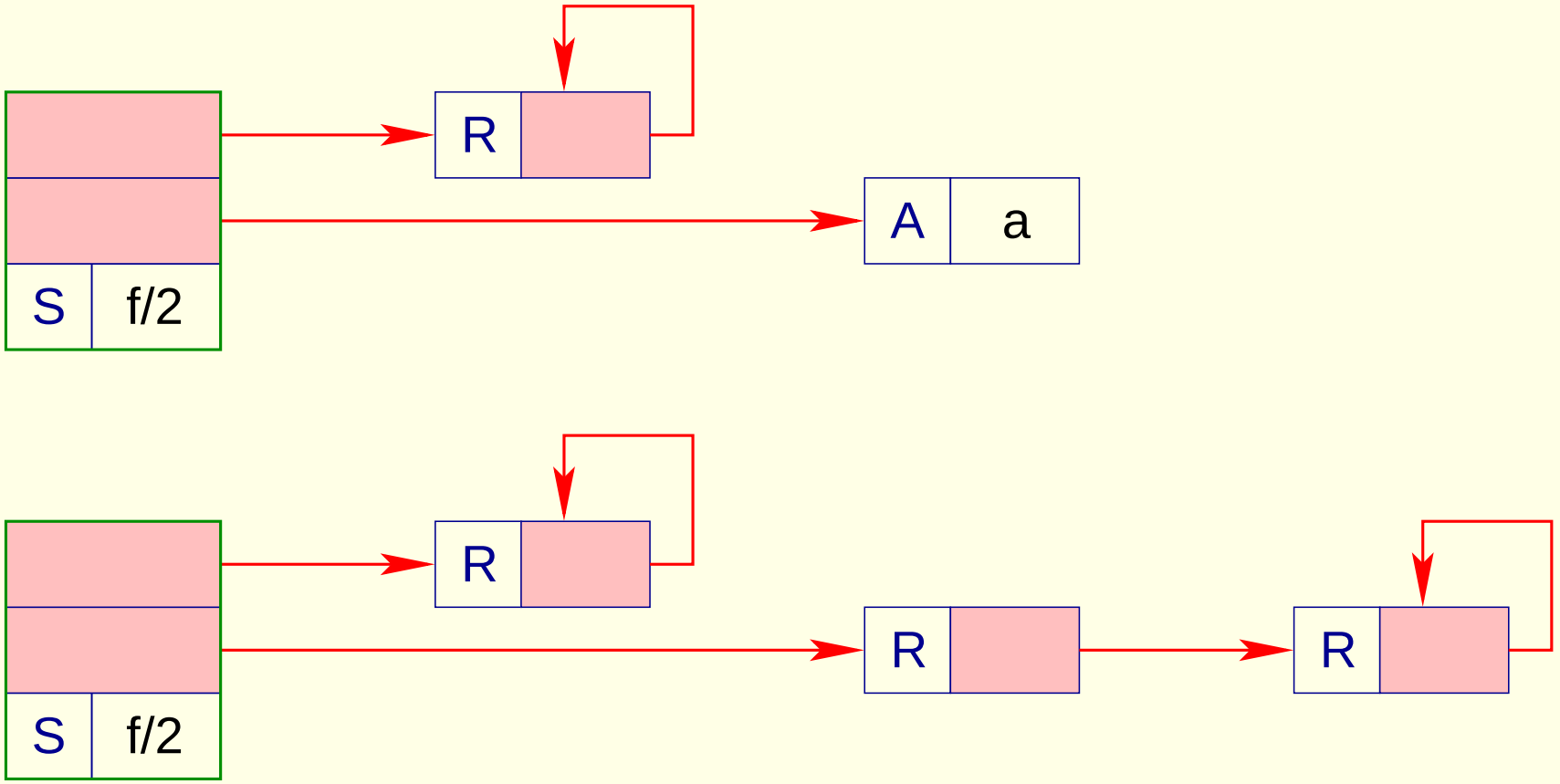
Unifitseerimine

```
...
  if (H[v] == (R,_)) {
    if (check (v,u)) {
      H[v] = (R,u); trail(v); return true;
    } else { backtrack(); return false; }
  }
  if (H[u] == (A,a) && H[v] == (A,a)) return true;
  if (H[u] == (S,f/n) && H[v] == (S,f/n)) {
    for (int i=1; i<=n; i++;)
      if (!unify (deref (H[u+i]), deref (H[v+i])) return false;
    return true;
  }
  backtrack(); return false;
}
```

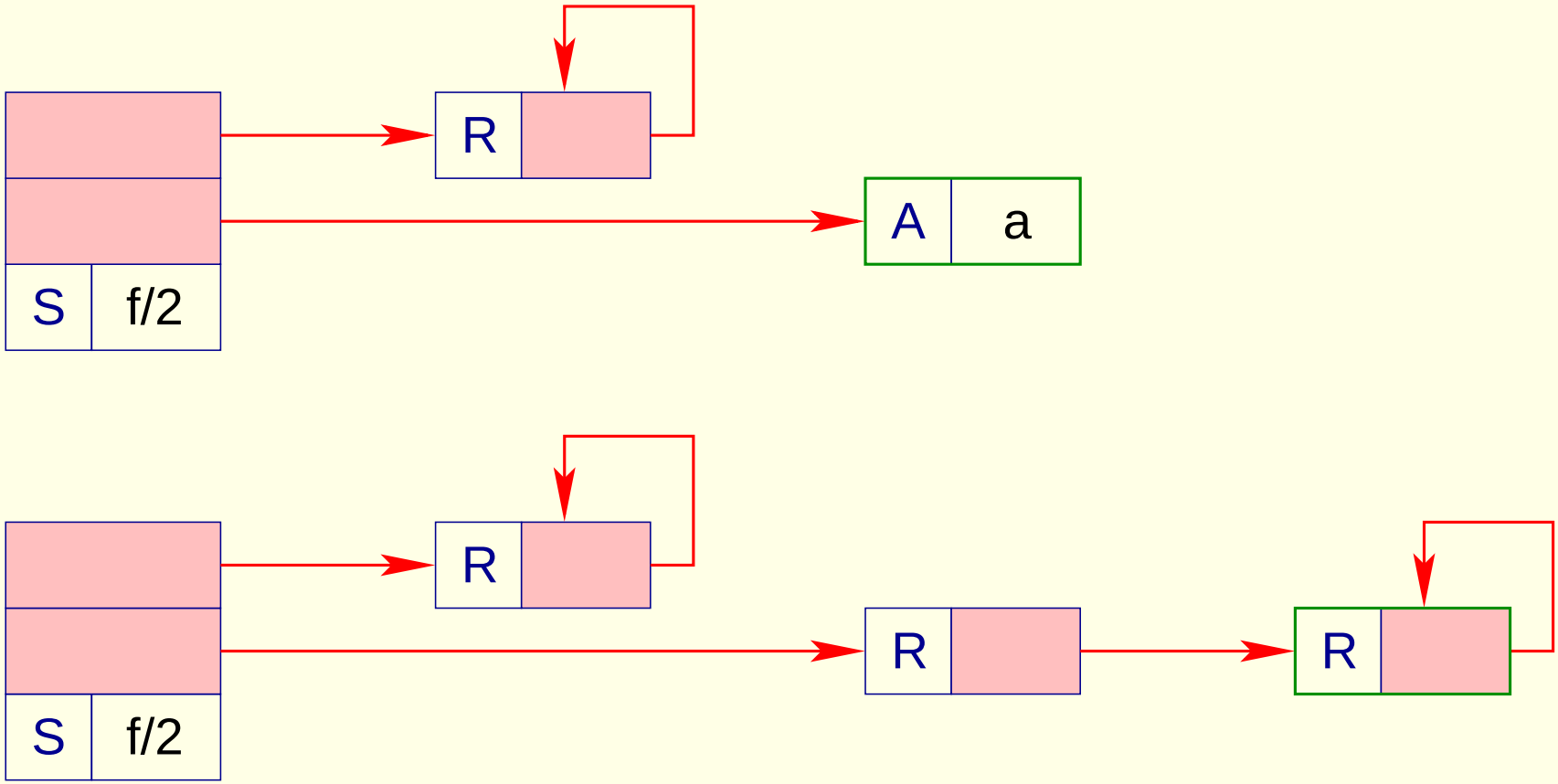
Unifitseerimine



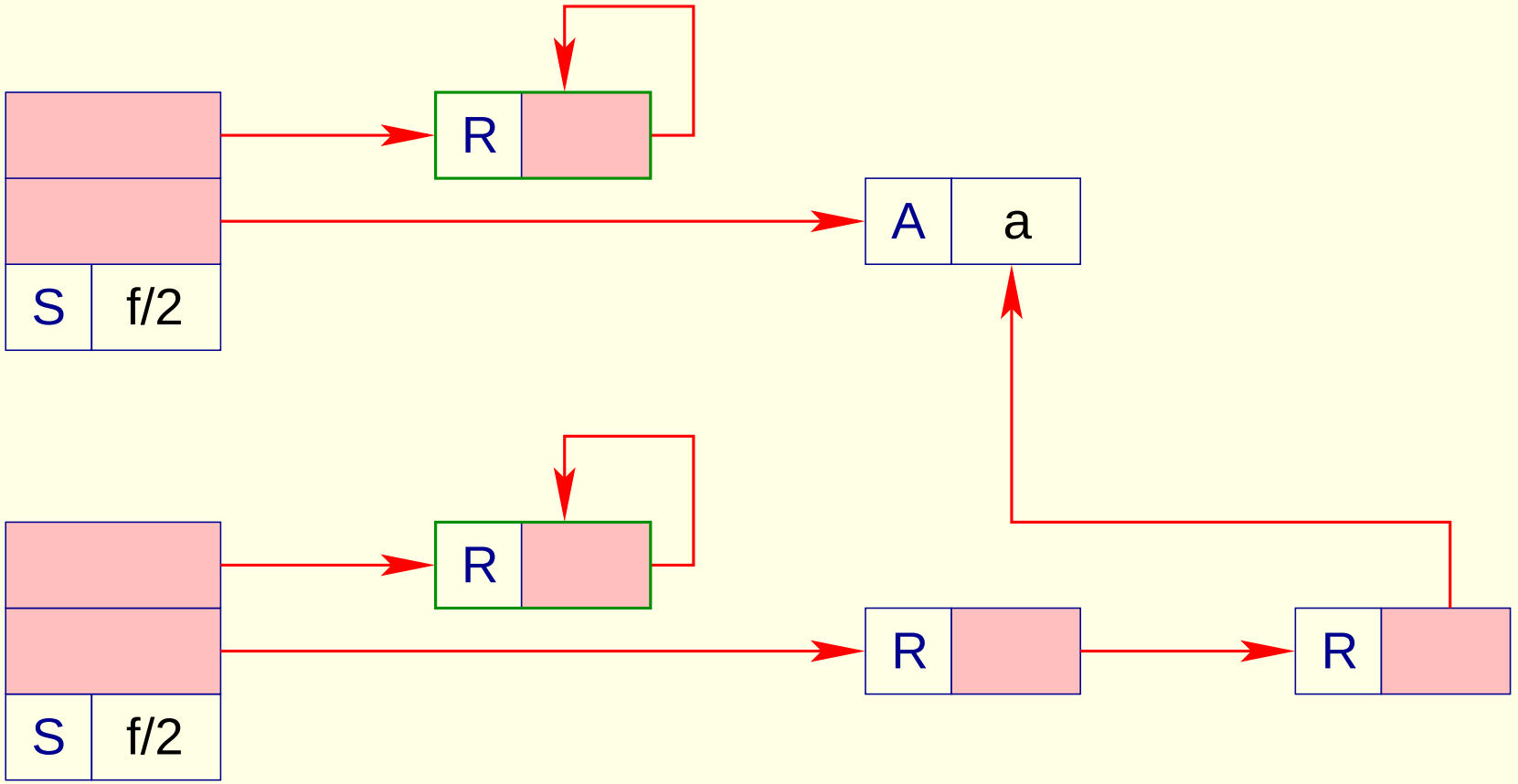
Unifitseerimine



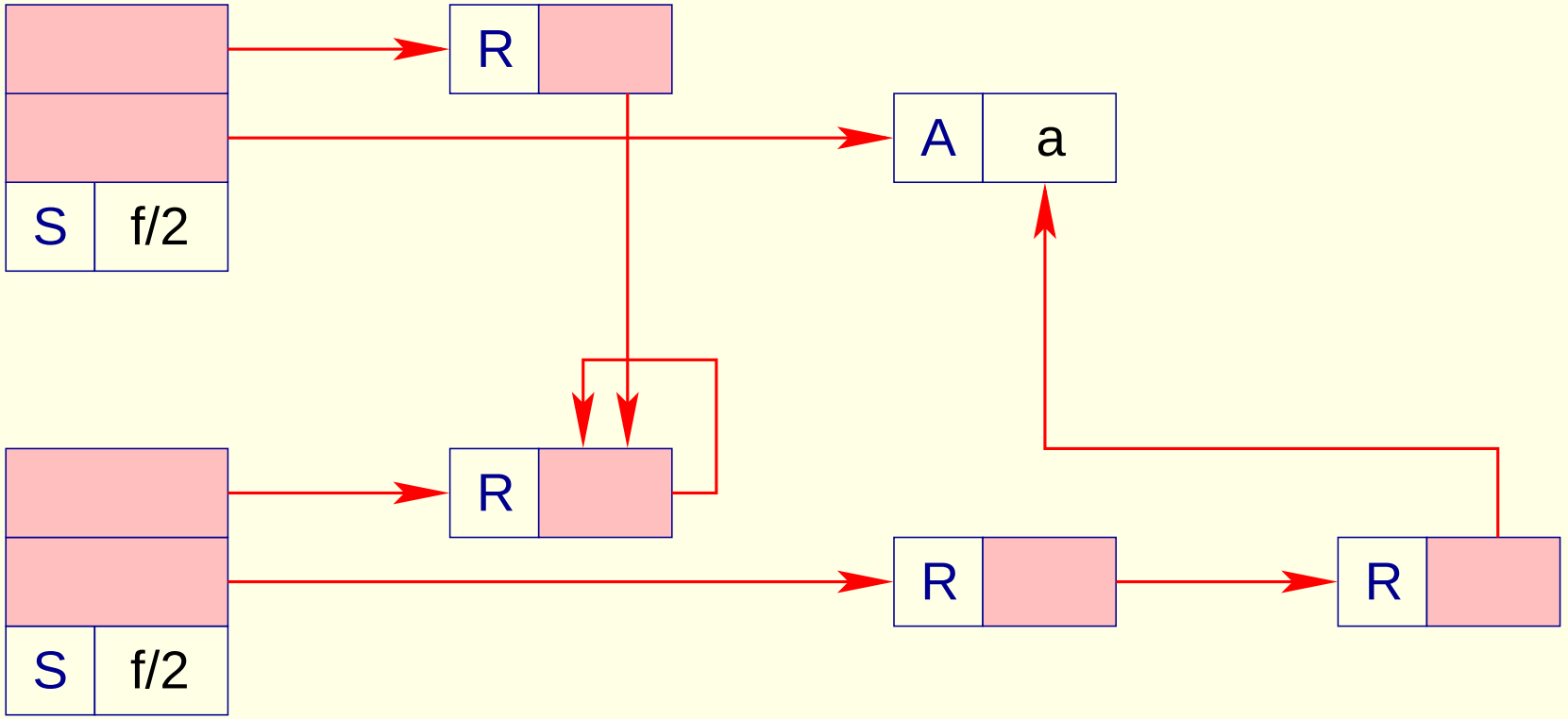
Unifitseerimine



Unifitseerimine



Unifitseerimine



Unifitseerimine

- Funktsioon `trail()` salvestab uued seosed.
- Funktsioon `backtrack()` teostab tagasipöördumise.
- Funktsioon `check()` kontrollib, kas muutuja (tema esimene argument) esineb termis (tema teine argument).
- Tihti seda kontrolli ei teostata:

```
bool check (ref u, ref v) {
    return true;
}
```

Unifitseerimine

Kui soovime kontrolli teostada, siis saame funktsiooni `check()` realiseerida järgnevalt:

```
bool check (ref u, ref v) {
  if (u == v) return false;
  if (H[v] == (S,f/n)) {
    for (int i=1; i<=n; i++)
      if (!check (u, deref (H[v+i])))
        return false;
  return true;
}
```

Unifitseerimine

- Võrrandi $\tilde{X} = t$ transleerimine on väga lihtne,
- aga tihti muutuvad konstrueeritud objektid koheselt *prügiks*.
- Idee:
  - Lisame muutujale $\tilde{X}$ vastava seose magasini.
  - Väldime termi t alamtermide konstrueerimist kui võimalik.
  - Selle asemel iga t tipu käsuks, mis teostab selle tipuga unifitseerimise!

$$\begin{aligned} \text{code}_G\ (\tilde{X} = t)\ \rho \quad &= \quad \text{put}\ \tilde{X}\ \rho \\ &\qquad\quad \text{code}_U\ t\ \rho \end{aligned}$$

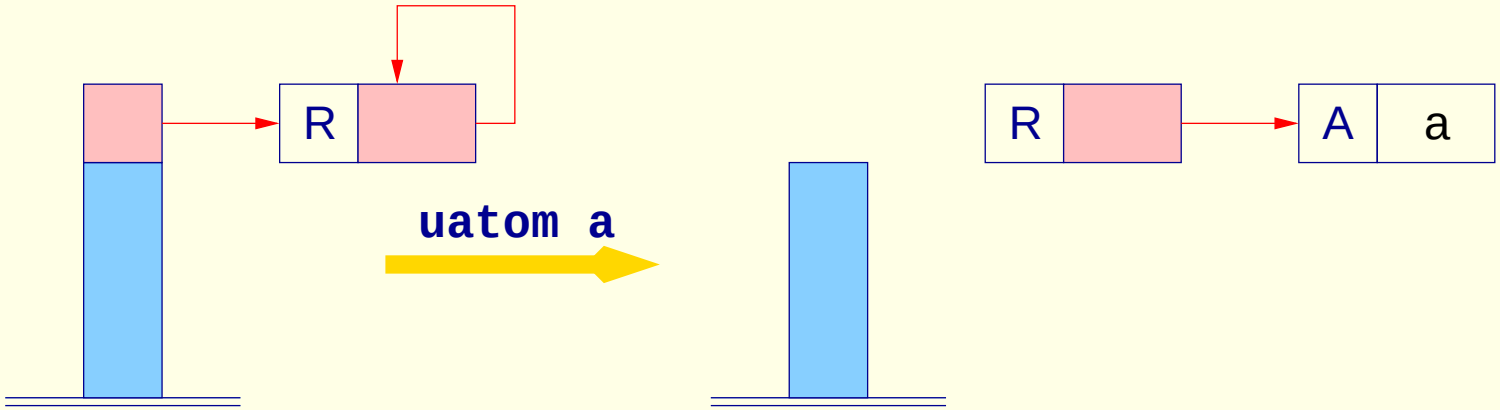
Unifitseerimine

Lihtsate termide unifitseerimine:

$$
\begin{aligned}
\text{code}_U\ a\ \rho &= \texttt{uatom a} \\
\text{code}_U\ X\ \rho &= \texttt{uvar}\ (\rho\ X) \\
\text{code}_U\ \overline{X}\ \rho &= \texttt{uref}\ (\rho\ X) \\
\text{code}_U\ \_\ \rho &= \texttt{pop}
\end{aligned}
$$

Unifitseerimine

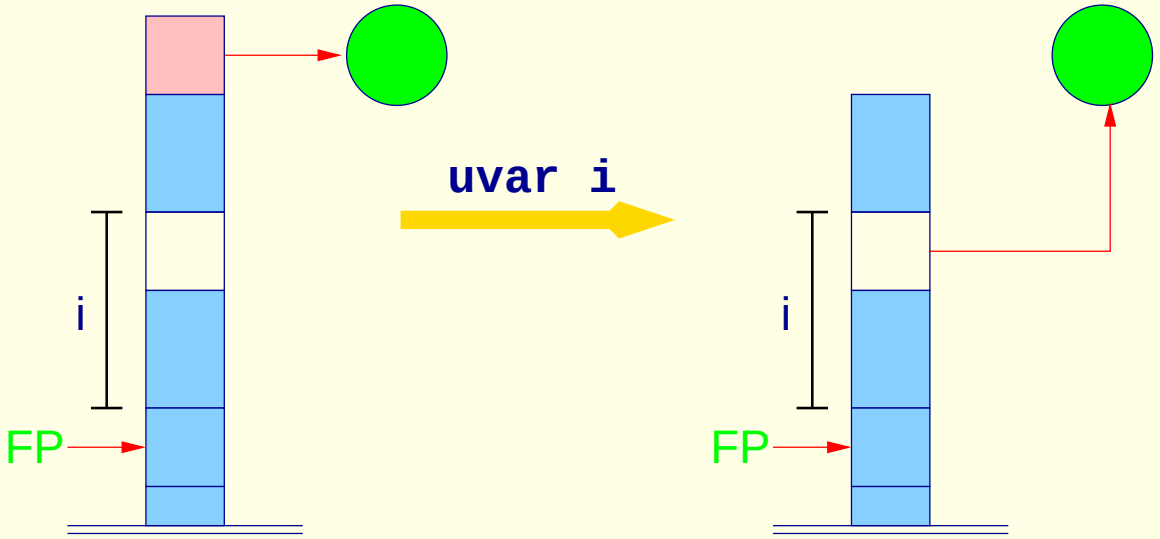
Käsk `uatom a` realiseerib unifitseerimise aatomiga




```
v = S[SP]; SP−−;
switch (H[v]) {
      case (A,a):  break;
      case (R,_):  H[v] = (R, new(A,a));
                   trail(v); break;
      default:     backtrack();
}
```

Unifitseerimine

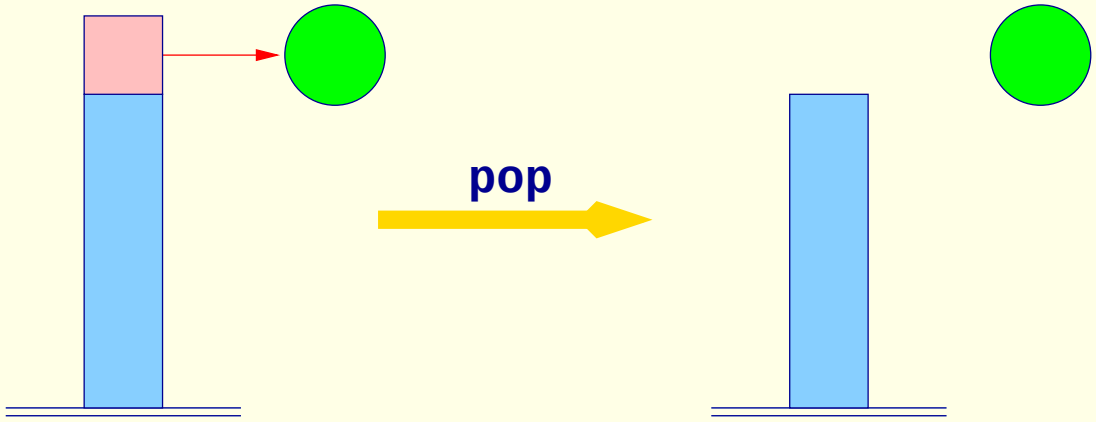
Käsk `uvar i` realiseerib unifitseerimise initsialiseerimata muutujaga



S[FP+i] = S[SP];
SP−−;

Unifitseerimine

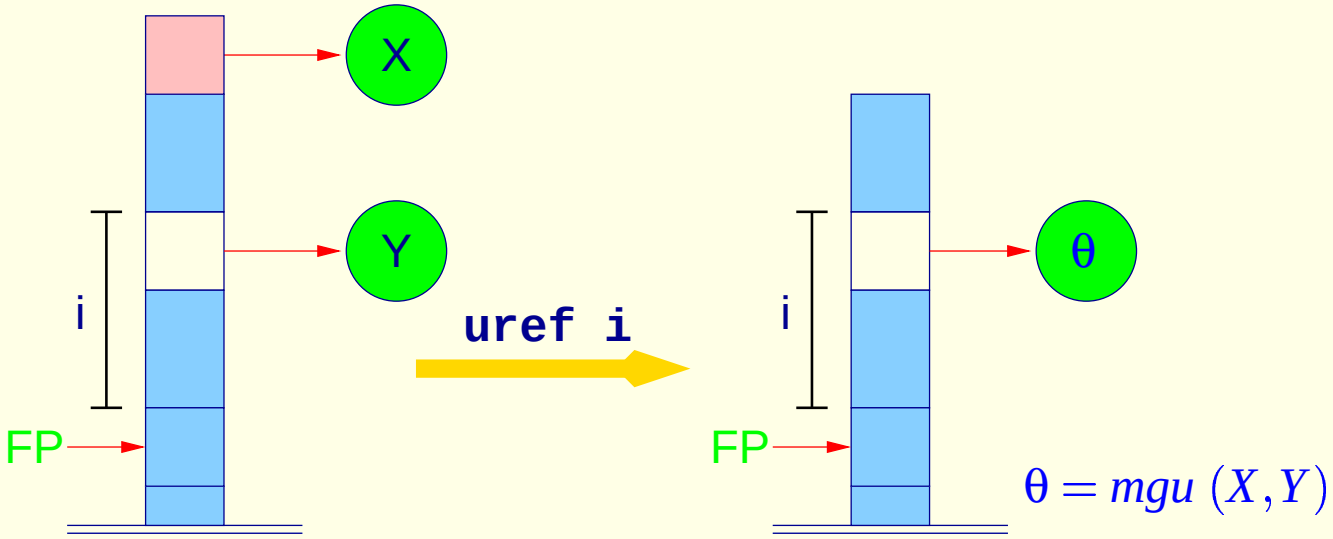
Käsk pop realiseerib unifitseerimise anonüümse muutujaga



SP−−;

Unifitseerimine

Käsk `uref i` realiseerib unifitseerimise initsialiseeritud muutujaga




```
unify (S[SP], deref(S[FP+i]));
SP−−;
```

Ainus koht, kus täitmisaegset funktsiooni `unify()` kasutatakse!

Unifitseerimine

Funktsiooni rakenduste unifitseerimine:

- Termiga *t* unifitseerimine toimub termi läbimisel *eeljärjekorras*.
- Kõigepealt kontrollime, kas tipmine viit on unifitseeruv term.
- Kui ta on sama funktsioonisümboli rakendus, siis rekursiivselt kontrollime alamterme.
- Initsialiseerimata muutuja korral lülitume kontrollimiselt ümber termi konstrueerimisele.

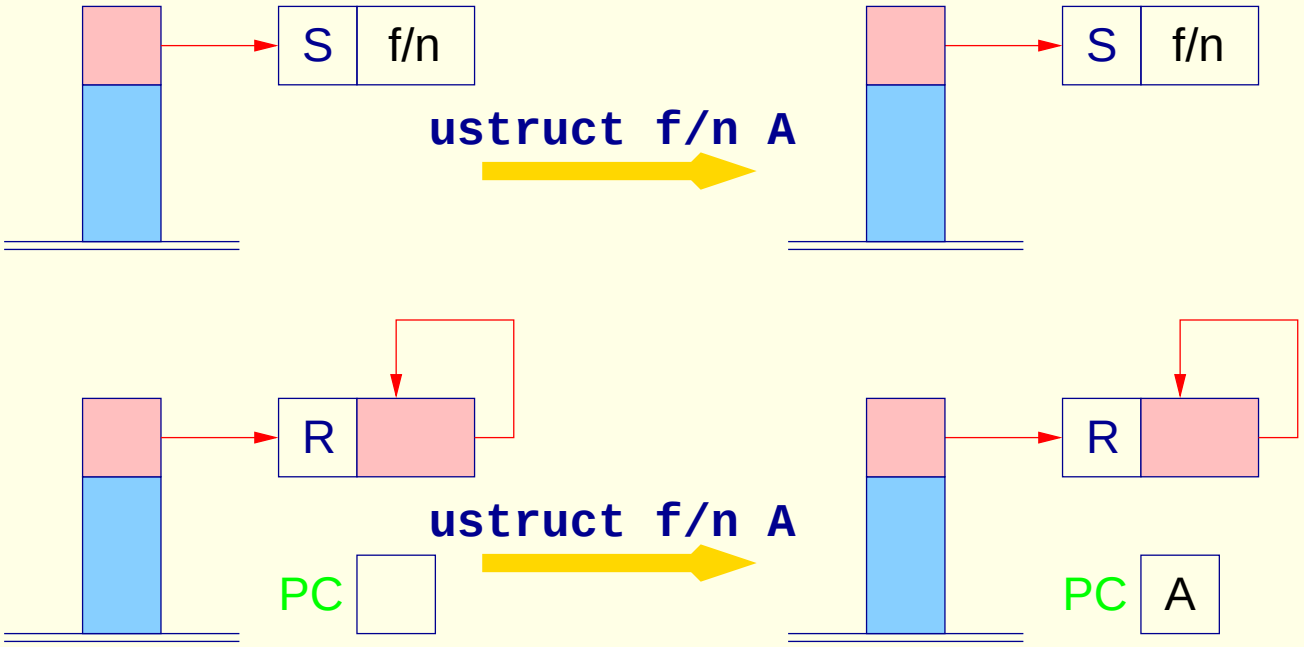
Unifitseerimine

Funktsiooni rakenduste unifitseerimine:

$\text{code}_U\ f(t_1,\ldots,t_n)\ \rho\ =$

```
    ustruct f/n A
    son 1
    code_U t_1 ρ
    ...
    son n
    code_U t_n ρ
    up B
A:  check ivars(f(t_1,...,t_n)) ρ
    code_A f(t_1,...,t_n) ρ
    bind
B:  ...
```

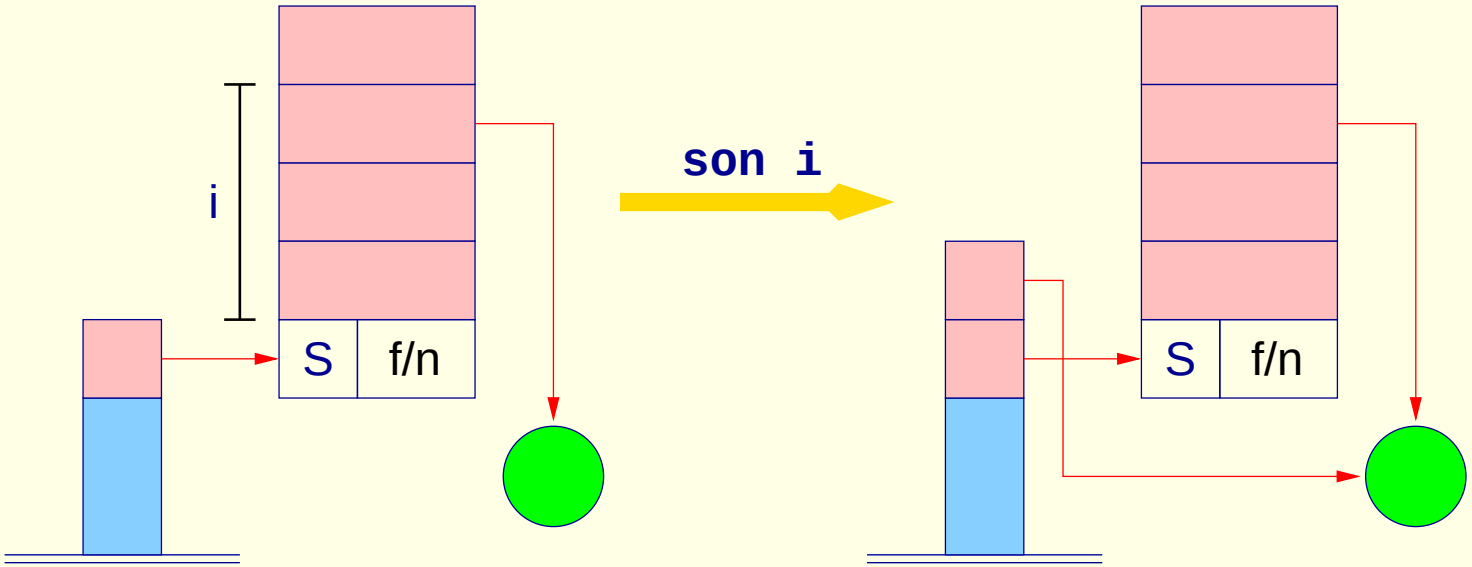
Unifitseerimine




```
switch (H[S[SP]]) {
      case (S,f/n):     break;
      case (R,_):       PC = A; break;
      default:          backtrack();
}
```

Unifitseerimine

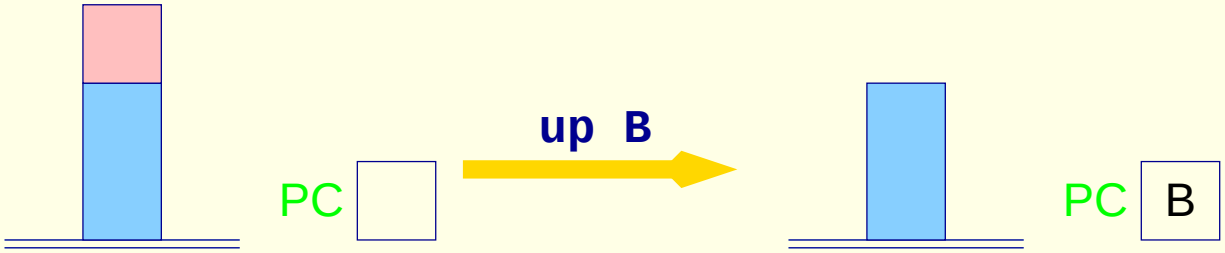
Käsk `son i` lisab i-nda alamtermi aadressi magasini tippu




```
S[SP+1] = deref(H[S[SP]+i]);
SP++;
```

Unifitseerimine

Käsk `up B` eemaldab magasinist ülemise pesa ja hüppab unifitseerimisele järgnevale koodile




```
SP−−;
PC = B;
```

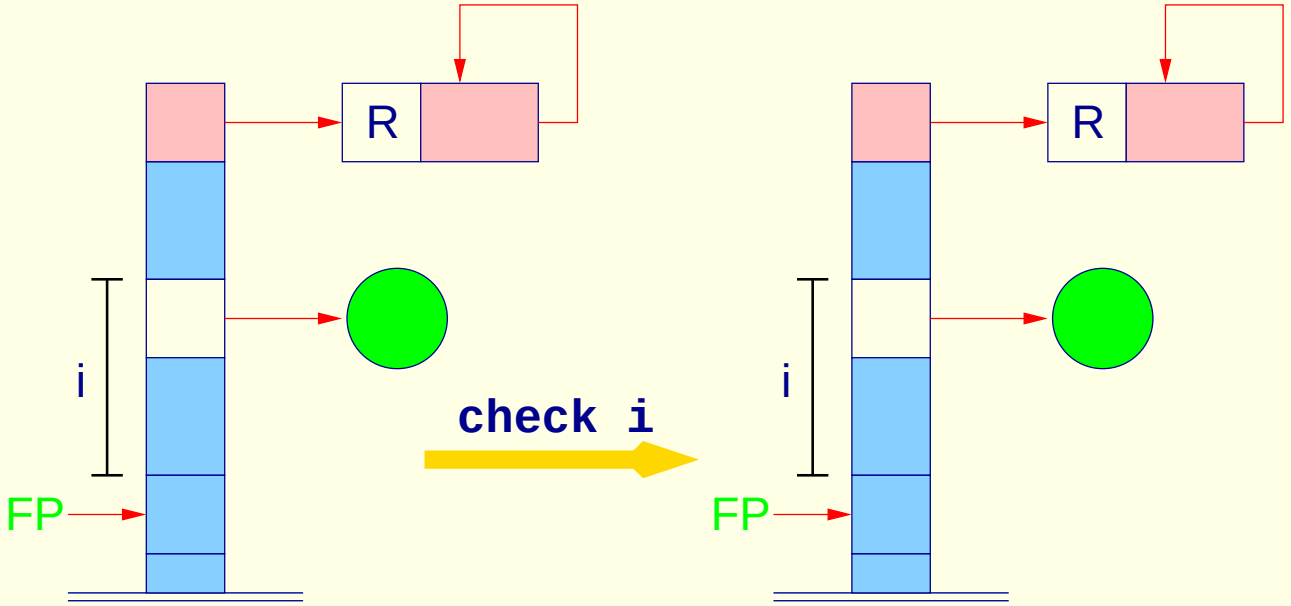
Unifitseerimine

- Initsialiseerimata muutuja korral lülitume kontrollimiselt ümber termi konstrueerimisele.
- Enne uue termi konstrueerimist peame veenduma, et ta ei sisalda magasini tipus olevat muutujat:
  - funktsioon ivars(t) väjastab termis t initsialiseerutud muutujate hulga;
  - makro check $\{Y_1, \ldots, Y_d\}$ ρ genereerib vajalikud testid:

$$\begin{aligned} \text{check } \{Y_1, \ldots, Y_d\}\ \rho \quad = \quad & \text{check } \rho\ Y_1 \\ & \ldots \\ & \text{check } \rho\ Y_d \end{aligned}$$

Unifitseerimine

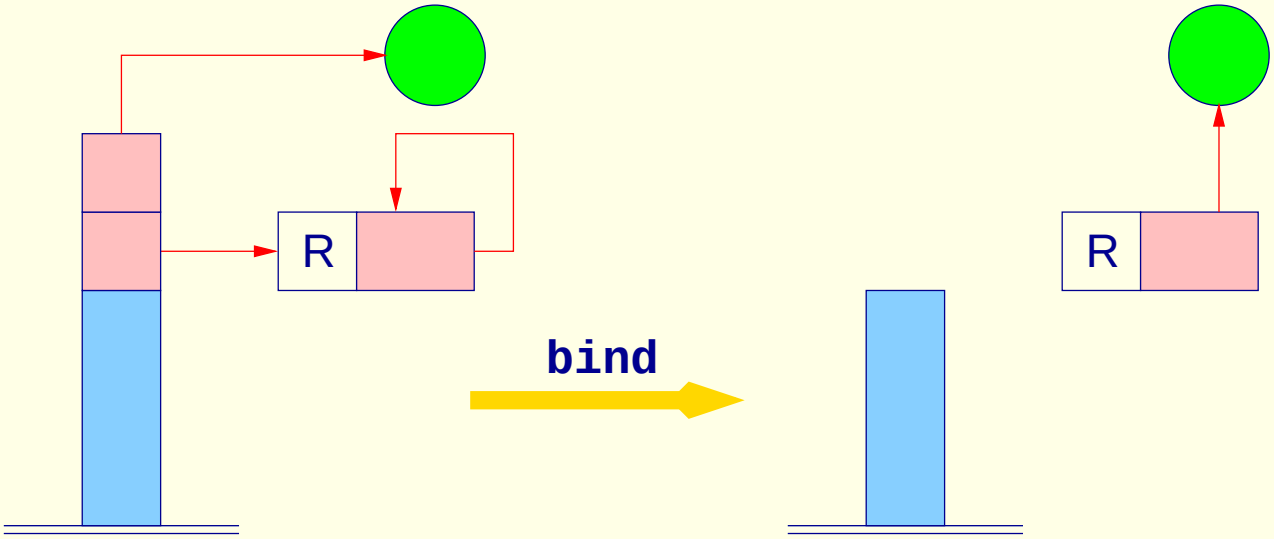
Käsk `check i` kontrollib, kas magasini tipus olev (initsialiseerimata) muutuja esineb i-nda muutujaga seotud termis




```
if (!check (S[SP], deref(S[FP+i]))
      backtrack();
```

Unifitseerimine

Käsk `bind` seob (initsialiseerimata) muutuja konstrueeritud termiga




```
H[S[SP−1]] = (R,S[SP]);
trail (S[SP−1]);
SP = SP − 2;
```

Unifitseerimine

Näide: olgu antud term $f(g(\bar{X}, Y), a, Z)$ ja keskkond $\rho = \{X \mapsto 1,\ Y \mapsto 2,\ Z \mapsto 3\}$:

```
ustruct f/3 A1      up B2          B2: son 2          putvar 2
son 1           A2: check 1            uatom a        putstruct g/2
ustruct g/2 A2      putref 1           son 3          putatom a
son 1               putvar 2           uvar 3         putvar 3
uref 1              putstruct g/2      up B1          putstruct f/3
son 2               bind           A1: check 1        bind
uvar 2                                 putref 1   B1: ...
```