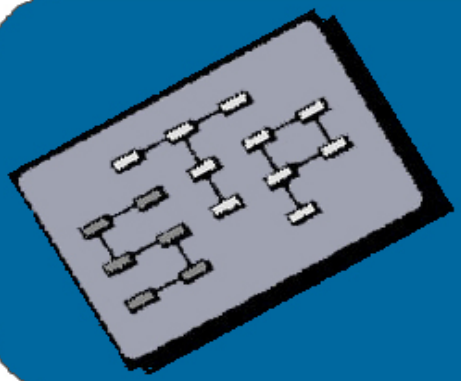
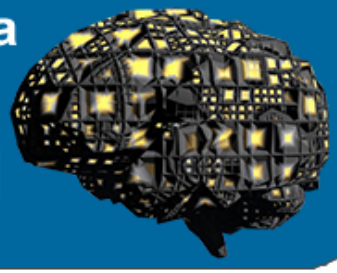


Datorzinātnes un informācijas tehnoloģijas fakultāte

Sistēmu teorijas un projektēšanas katedra

MĀKSLĪGĀ INTELEKTA PAMATI



5. Modulis "Mākslīgā intelekta loģiskie pamati"

5.5. Tēma

Loģiskās secināšanas nodrošināšana

Dr.habil.sc.ing., profesors **Jānis Grundspeņķis**, Dr.sc.ing., lektore **Alla Anohina**

Sistēmu teorijas un projektēšanas katedra
Datorzinātnes un informācijas tehnoloģijas fakultāte
Rīgas Tehniskā universitāte
E-pasts: *{janis.grundspenkis, alla.anohina}@rtu.lv*
Kontaktadrese: *Meža iela 1/4- {550, 545}, Rīga, Latvija, LV-1048*
Tālrunis: *(+371) 67089{581, 595}*

Tēmas mērķi un uzdevumi

Tēmas mērķis ir sniegt zināšanas par loģiskās secināšanas nodrošināšanu intelektuālās sistēmās.

Pēc šīs tēmas apgūšanas Jūs:

- pratīsiet pārrakstīt teikumus secināšanas likumiem vajadzīgajā formā;
- zināsiet substitūciju likumus un substitūciju kompozīcijas būtību;
- zināsiet, kas ir unifikācija, un pratīsiet to demonstrēt.

Secināšanas nodrošināšana (1)

Kā jau bija minēts, secināšana balstās tikai un vienīgi uz teikumu sintaksi. Tas nozīmē, ka pielietojot secināšanas likumu, sistēmai ir jāspēj noteikt, vai divi teikumi sakrīt.

Izteikumu rēķinos tas ir pietiekami vienkārši. Sakarā ar to, ka vienam teikumam atbilst viens simbols, sistēmai ir jāatrod vienādi simboli

Secināšanas nodrošināšana (2)

Predikātu loģikā noteikt, vai divi teikumi sakrīt, ir sarežģītāk, jo teikumos tiek izmantoti mainīgie:

bokseris(X)

bokseris(ingus)

} Divas dažādas simbolu virknes, ja ņemt vērā tikai pašus simbolus

Dotajā piemērā sistēmai ir jāzina, ka X var aizvietot ar *ingus*. Šim nolūkam izmanto *unifikāciju*.

Ar ***unifikāciju*** saprot algoritmu, kas nepieciešams, lai noteiktu substitūcijas, kuras ir vajadzīgas, lai divi predikātu rēķinu teikumi kļūtu vienādi.

Secināšanas nodrošināšana (3)

Secināšanas likumiem ir vajadzīga noteikta likumu forma. To var panākt šādā veidā:

- Jāizslēdz universālkvantori, jo teikums ir patiess jebkurai mainīgo piesaistei
- Jāizslēdz eksistenciāli kvantificētie mainīgie no zināšanu bāzes
- Jāpielieto unifikācija, lai iegūtu likumu formu, kas ir vajadzīga secināšanas likumiem

Eksistenciāli kvantificētu mainīgo izslēgšana (1)

Eksistenciāli kvantificētus mainīgos var izslēgt šādos veidos:

- Mainīgos aizvieto ar konstantēm, kas padara teikumu par patiesu

∃ **X vecāks (X,jānis)**

var aizvietot ar

vecāks(ivars, jānis) vai **vecāks(marija, jānis),**

ja Ivars un Marija ir Jāņa vecāki dotajā interpretācijā

Eksistenciāli kvantificētu mainīgo izslēgšana (2)

- Pielieto *skolemizāciju*, ja mainīgā vērtība ir atkarīga no citu izteiksmē esošo mainīgo vērtībām

$$\forall \mathbf{X}\ \exists \mathbf{Y}\ \textbf{māte (X,Y)}$$

Eksistenciāli kvantificēts mainīgais Y ir atkarīgs no X

Skolemizācija aizvieto katru eksistenciāli kvantificētu mainīgo ar funkciju, kas dod piemērotu konstanti kā citu teikumā esošo mainīgo funkciju

$$\forall \mathbf{X}\ \textbf{māte (X, f(X))}$$

f(X) ir Skolema funkcija, kas aizvieto atkarīgo mainīgo

Novērtējot funkciju, iegūst tās vērtību-konstanti, kas, savukārt, ļauj izslēgt eksistences kvantoru

Eksistenciāli kvantificētu mainīgo izslēgšana (3)

Piemērs:

"Ikvienam ir smadzenes"

$\forall$X persona(X) -> $\exists$Y smadzenes (Y) $\wedge$ pieder (X, Y)

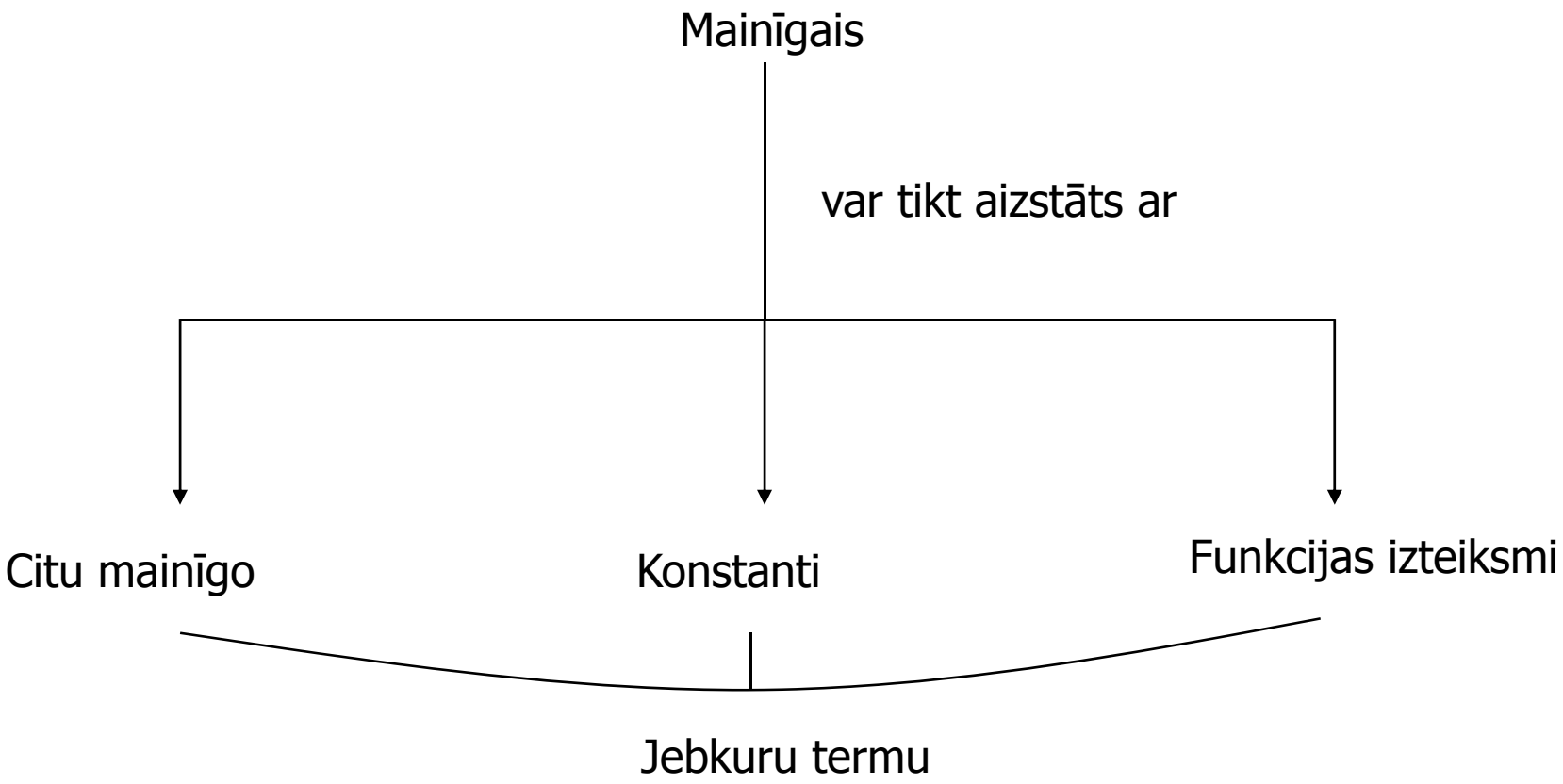
Ja Y aizvietot ar konstanti s, tiks iegūts teikums

$\forall$X persona(X) -> smadzenes (s) $\wedge$ pieder (X, s)

kas nozīmē, ka visiem ir vienāda smadzeņu kopija. Taču ir jāpasaka, ka smadzenes nav kopijas

$\forall$X persona(X) -> smadzenes (f(X)) $\wedge$ pieder (X, f(X))

Secināšanas nodrošināšana

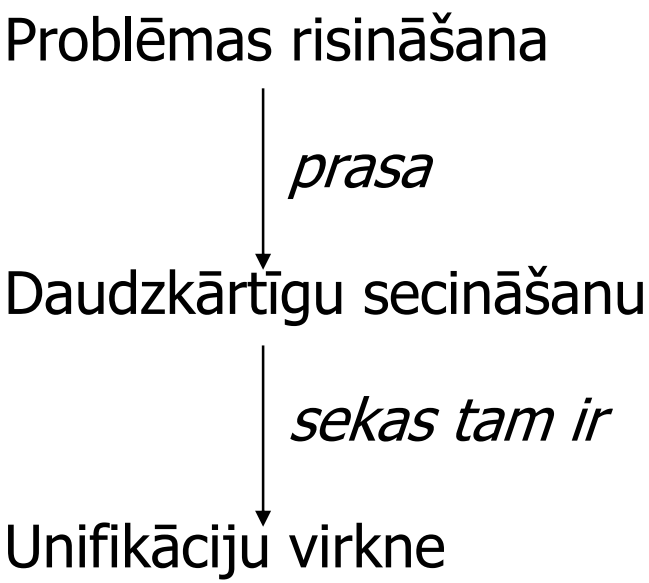


Substitūciju likumi

Substitūciju likumi:

- Katru mainīgo var sistemātiski aizvietot ar konstanti
- Katra konstante ir pamateksemplārs un to nevar aizvietot
- Vienu mainīgo nevar aizvietot ar divām dažādām konstantēm
- Ja mainīgajam piesaista konstanti, tad šim mainīgajam vairs nevar dot jaunu piesaisti
- Mainīgo var aizvietot ar citu mainīgo
- Mainīgo var aizvietot ar funkcijas izteiksmi, ja tā nesatur šo mainīgo
- Mainīgo nevar aizvietot ar termu, kas satur šo mainīgo

Unifikāciju virkne



Mainīgo unifikācijai ir jābūt saskaņotai. Ja mainīgajam ir piesaiste, tad visām pārējām unifikācijām un secināšanai tas ir jāņem vērā.

Pilna substitūciju kopa

Pilna substitūciju kopa ir svarīga secināšanas ķēdēs, jo tā var saturēt atbildi uz oriģinālo jautājumu.

p(X,Y)->q(X,Y)

p(a,Z)

Substitūcija {a/X, Z/Y}

p(a,Z)->q(a,Z)
p(a,Z)
q(a,Z)

q(W,b)->s(W,B)
Substitūcija {a/W, b/Z}

q(a,b)->s(a,b)
q(a,b)
s(a,b) Atbilde

Substitūciju kompozīcija

Substitūciju kompozīcija: Ja S un S` ir 2 substitūciju kopas, tad S un S` kompozīcija (SºS`) tiek iegūta, pielietojot S` kopas S elementiem un pievienojot rezultātu kopai S.

{X/Y, W/Z}

{V/X}

{a/V, f(b)/W}

{X/Y, W/Z} º {V/X} → {V/ Y, W/ Z}

{V/ Y, W/ Z} º {a/V, f(b)/W} → {a/Y, f(b)/Z}

Substitūciju kompozīcija ir metode, ar kuru tiek apvienotas unifikācijas rezultātā iegūtās substitūcijas.

Visvispārīgākais unifikators

Unifikācijas algoritms prasa, lai unifikators būtu maksimāli vispārīgs, t.i. jebkuriem diviem teikumiem ir jāatrod visvispārīgākais unifikators.

Piemēram, teikumus p(X) un p(Y) var unificēt ar jebkuru konstanti. Taču izmantojot kā unifikatoru mainīgo, var iegūt vispārīgāku teikumu. Risinājumi, kas tiks iegūti, izmantojot konstanti, vienmēr ierobežos loģiskos izvedumus.

Ja s ir patvaļīgs teikuma T unifikators, bet g ir visvispārīgākais šīs teikumu kopas unifikators, tad pielietojot s teikumam T, eksistēs vēl viens unifikators r tāds, ka Ts=Tgr, kur Ts un Tgr ir unifikāciju kompozīcijas, kas ir pielietotas teikumam T.

Unifikācijas funkcija (1)

```
Function unify (E1, E2);
begin
  case1: abas izteiksmes E1 un E2 ir konstantes vai tukši saraksti
 1          if E1 = E2 then return ()
 2           else return (Neveiksme);
  case2: izteiksme E1 ir mainīgais
3           if E1 ietilpst izteiksmē E2 then return (Neveiksme)
4           else return (E2/E1);
  case3: izteiksme E2 ir mainīgais
5           if E2 ietilpst izteiksmē E1 then return (Neveiksme)
6           else return (E1/E2);
  case4: viena no izteiksmēm E1 vai E2 ir tukšs saraksts
7           return (Neveiksme);
 8 otherwise
    begin
9       HE1 := pirmais elements no E1;
10       HE2 := pirmais elements no E2;
11       SUBS1:= unify (HE1, HE2);
12       if SUBS1 = Neveiksme then return (Neveiksme);
13       TE1 := SUBS1 pielietošana atlikušajai E1 daļai;
14       TE2 := SUBS1 pielietošana atlikušajai E2 daļai;
15       SUBS2 := unify (TE1, TE2);
16       if SUBS2 = Neveiksme then return (Neveiksme)
17        else return (SUBS1 un SUBS2 kompozīcija);
    end;
  end of case;
end;
```

Unifikācijas funkcija (2)

Funkcija *unify* izskaitļo divu predikātu rēķinu teikumu substitūciju unifikācijas. Tā saņem kā parametrus divas predikātu rēķinu izteiksmes un atgriež vai nu visvispārīgāko unifikācijas substitūciju, vai nu konstanti *Neveiksme*, ja unifikācija nav iespējama. Šī funkcija ir rekursīva. Sākumā tā mēģina rekursīvi unificēt izteiksmju sākotnējās komponentes. Ja tas izdodas, visas substitūcijas, kas tiek iegūtas šīs unifikācijas rezultātā, tiek pielietotas izteiksmju atlikušajām daļām. Rekursija beidzas, kad par parametru kļūst simbols (predikāts, funkcija, konstante vai mainīgais), vai arī kad visi izteiksmes elementi kļūst saskaņoti.

Unifikācijas piemērs (1)

Piemērs:

Unifikācijas pielietošanas piemērs

Ir 2 izteiksmes predikātu rēķinos
basketbolists(X, tēvs(X))
basketbolists(biedriņš, tēvs(biedriņš))

Sarakstu veidā šīs izteiksmes izskatās šādi:
E1 (basketbolists X (tēvs X))
E2 (basketbolists biedriņš (tēvs biedriņš))

Unifikācijas pieleitošana:
unify((**basketbolists X (tēvs X)), (basketbolists biedriņš (tēvs biedriņš))**)
8. Rinda nostrādā
HE1:=basketbolists;
HE2:=basketbolists;
SUBS1:=unify(basketbolists,basketbolists);
case1, 1.rinda return()
TE1:=(X (tēvs X));
TE2:=(biedriņš (tēvs biedriņš));
SUBS2:=unify((X (tēvs X)), (biedriņš (tēvs biedriņš)));
8.rinda nostrādā
HE1:=X
HE2:=biedriņš;
SUBS1:=unify(X,biedriņš);
case2, 4.rinda return (biedriņš/X)

Unifikācijas piemērs (2)

Piemērs:

Unifikācijas pielietošanas piemērs (turp.)

TE1:=(tēvs biedriņš);
TE2:=(tēvs biedriņš);
SUBS2:=unify((tēvs biedriņš), (tēvs biedriņš));
8.rinda nostrādā
HE1:=tēvs;
HE2:=tēvs;
SUBS1:=unify(tēvs,tēvs);
case1, 1.rinda return ()
TE1:=(biedriņš);
TE2:=(biedriņš);
SUBS2:=unify((biedriņš), (biedriņš));
8.rinda nostrādā
HE1:=biedriņš;
HE2:=biedriņš;
SUBS1:=unify(biedriņš, biedriņš);
case1, 1.rinda return ()
Substitūciju kopa: {biedriņš/X}

Izveduma piemērs (1)

Piemērs:

Apskatīsim piemēru, kurā izvedums tiks realizēts, pielietojot Modus Ponens secināšanas likumu un veicot teikumu unifikāciju.

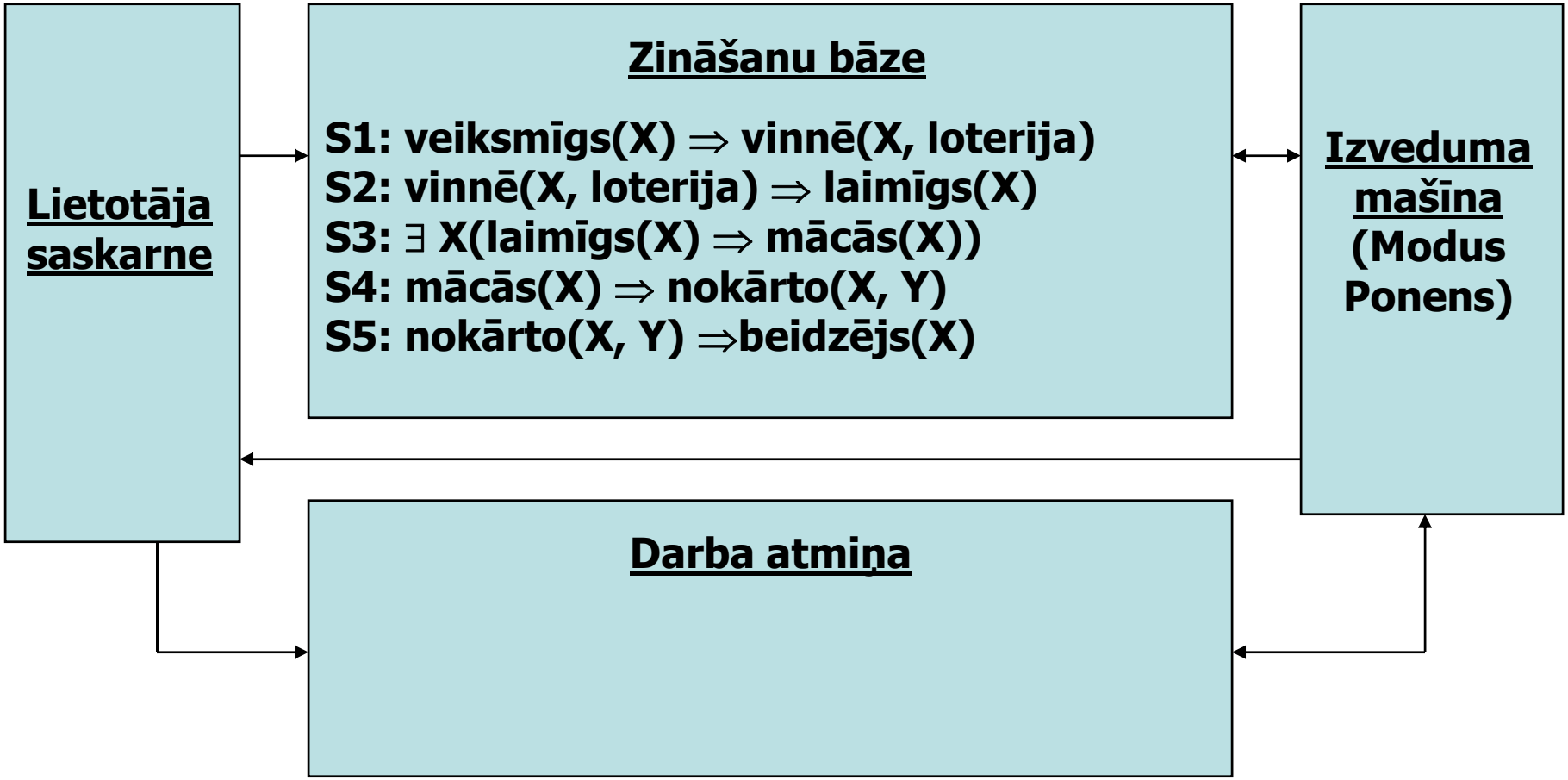
Problēmas apraksts:
1. Jebkurš, kas ir veiksmīgs, vinnē loterijā
2. Jebkurš, kas vinnē loterijā, ir laimīgs
3. Vismaz viens, kas ir laimīgs, mācās universitātē
4. Jebkurš, kas mācās universitātē, var nokārtot visus eksāmenus
5. Jebkurš, kas ir nokārtojis visus eksāmenus, beidz universitāti

Šos teikumus zināšanu bāzē var atspoguļot šādā veidā:
$\forall$ X (veiksmīgs(X) $\Rightarrow$ vinnē(X, loterija))
$\forall$ X (vinnē(X, loterija) $\Rightarrow$ laimīgs(X))
$\exists$ X (laimīgs(X) $\Rightarrow$ mācās(X))
$\forall$ X $\forall$ Y(mācās(X) $\Rightarrow$ nokārto(X, Y))
$\forall$ X $\forall$ Y(nokārto(X, Y) $\Rightarrow$ beidzējs(X))

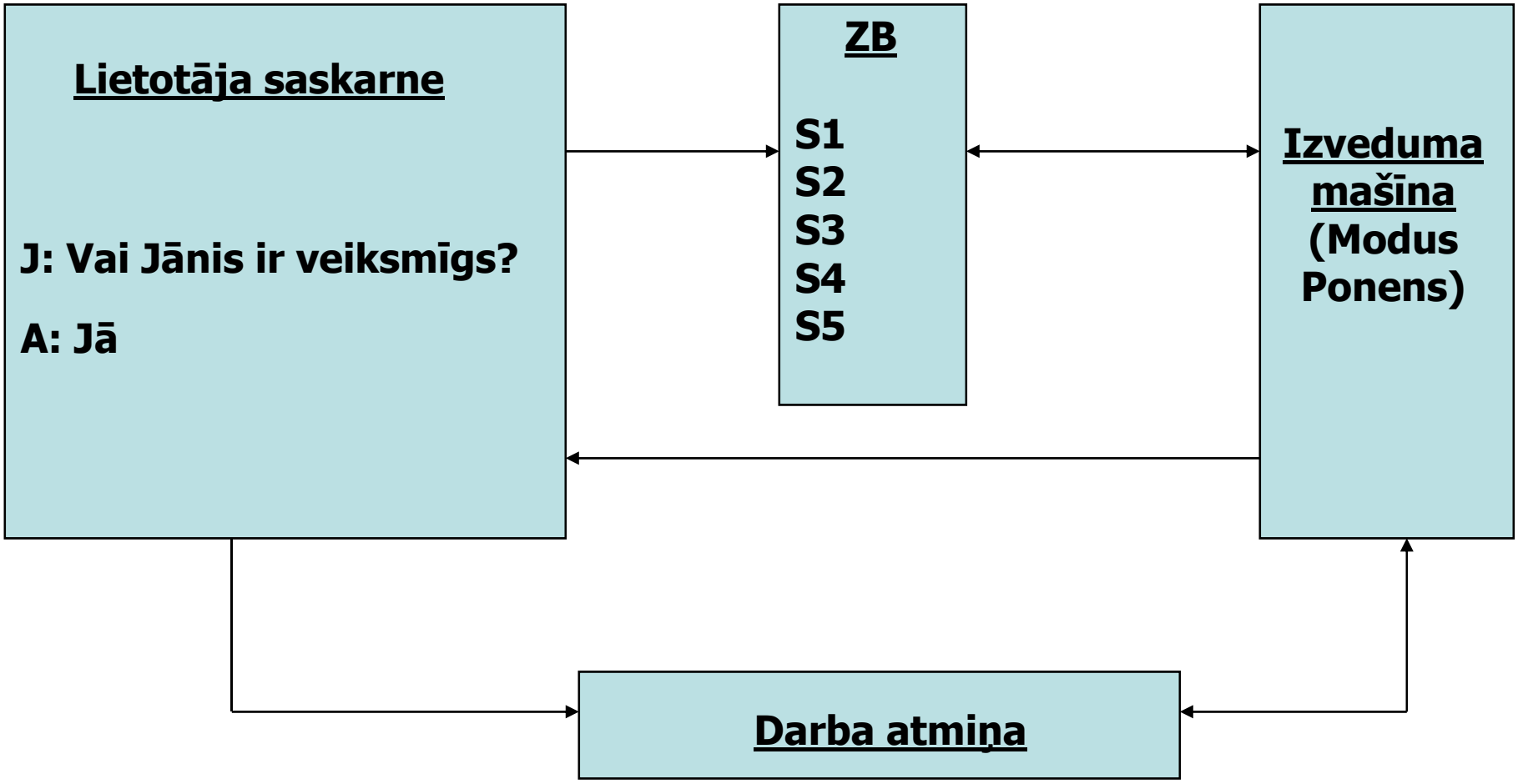
Pieņemsim, ir zināmi šādi fakti:

Jānis ir veiksmīgs	veiksmīgs(jānis)
Jānis nokārtoja MIP	nokārto(jānis,mip)

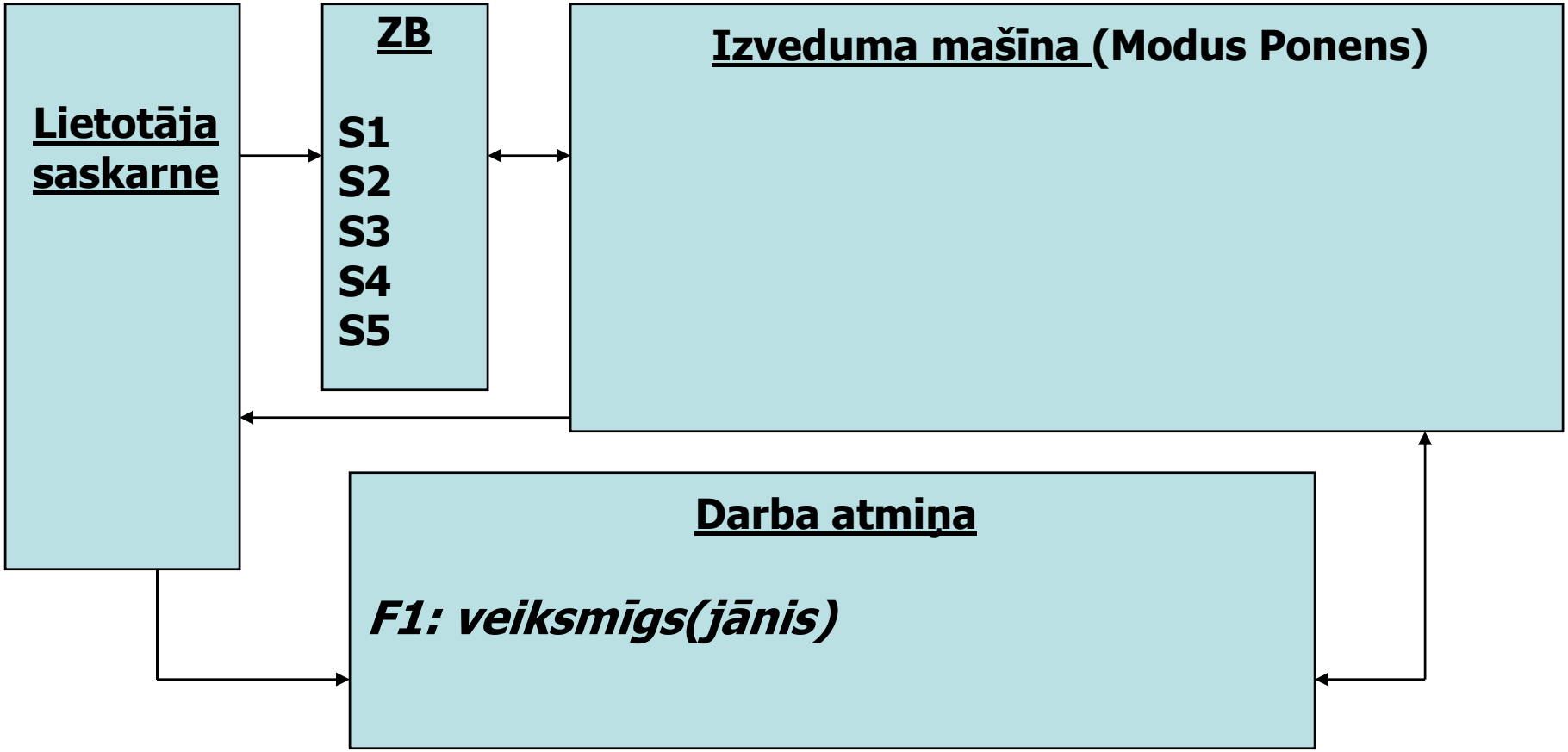
Izveduma piemērs (2)



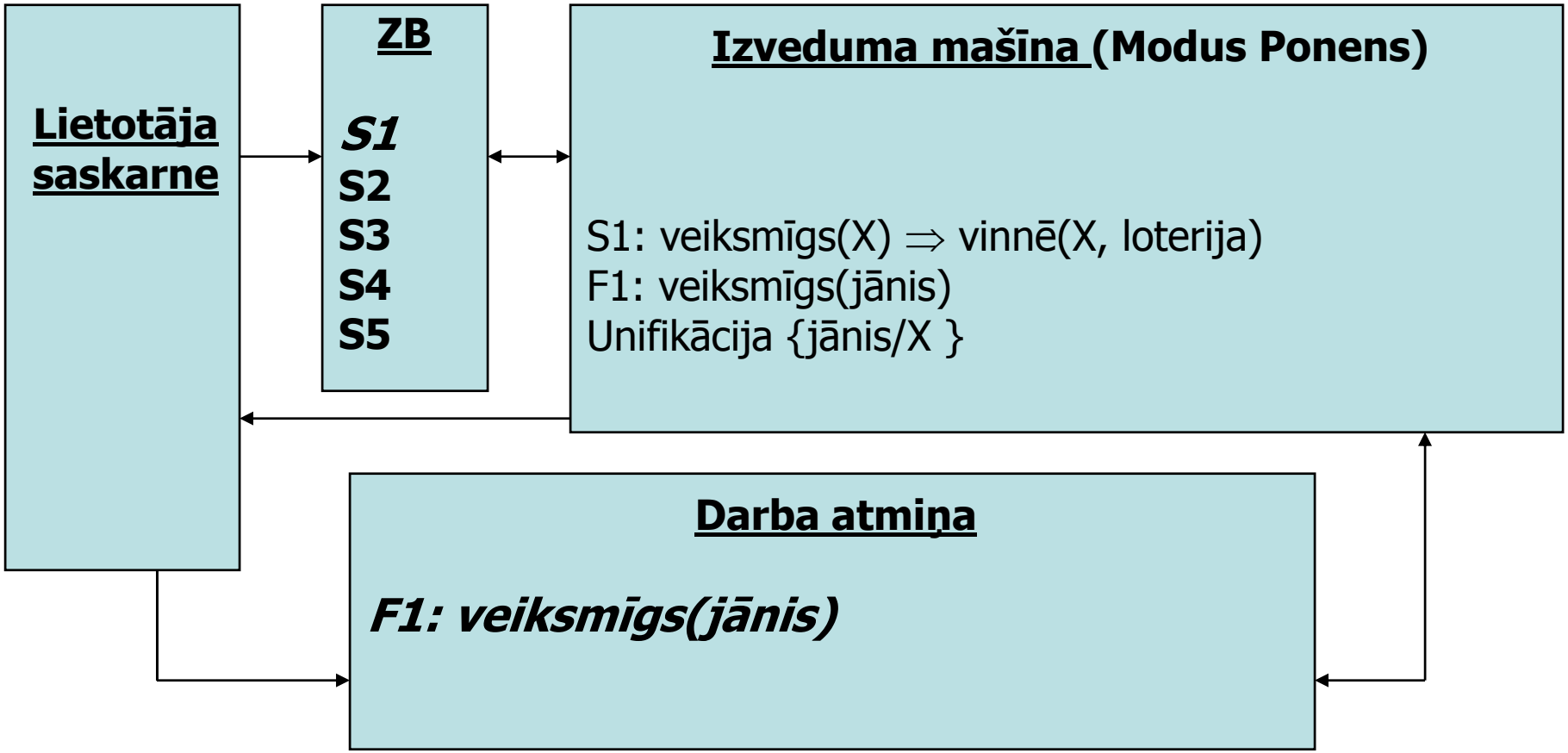
Izveduma piemērs (3)



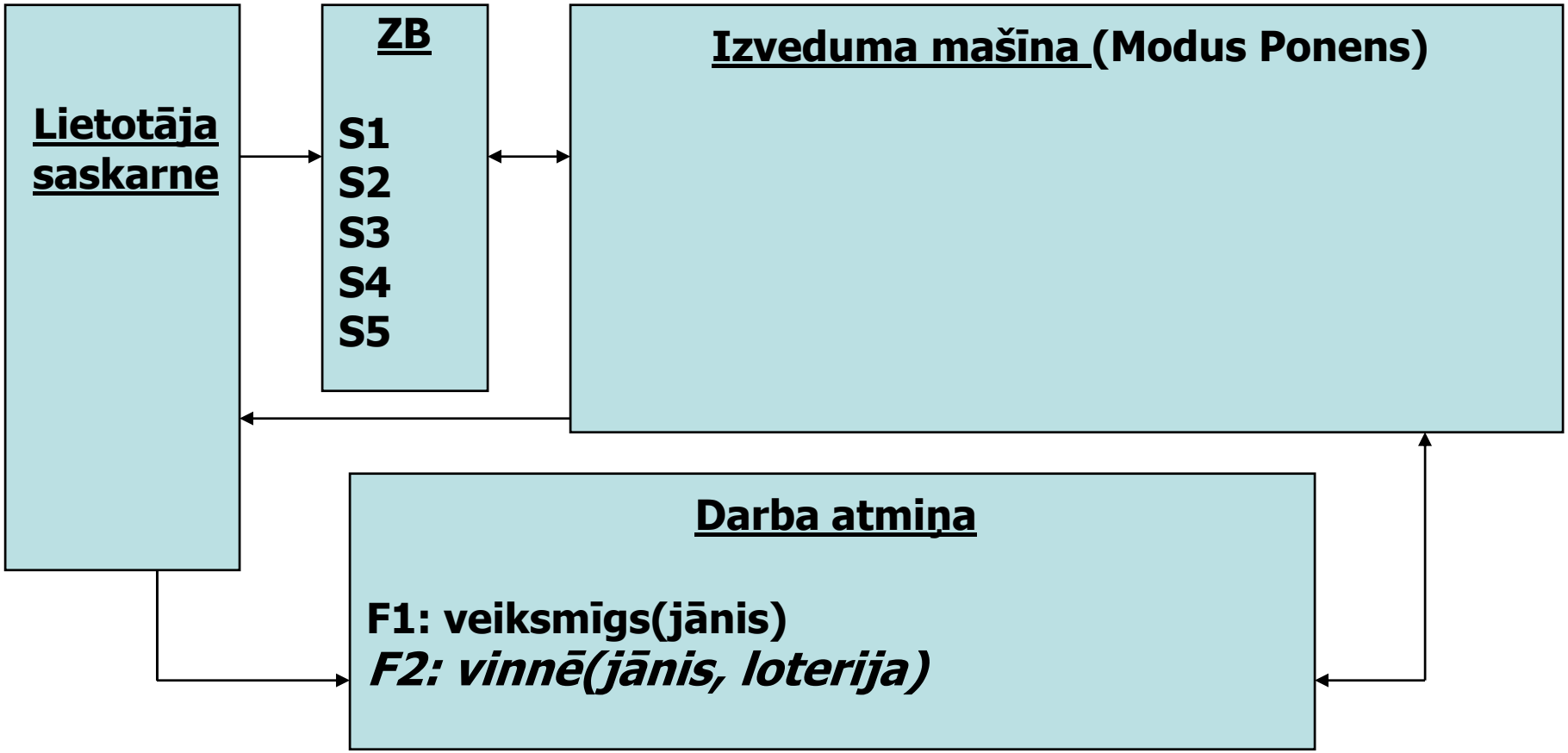
Izveduma piemērs (4)



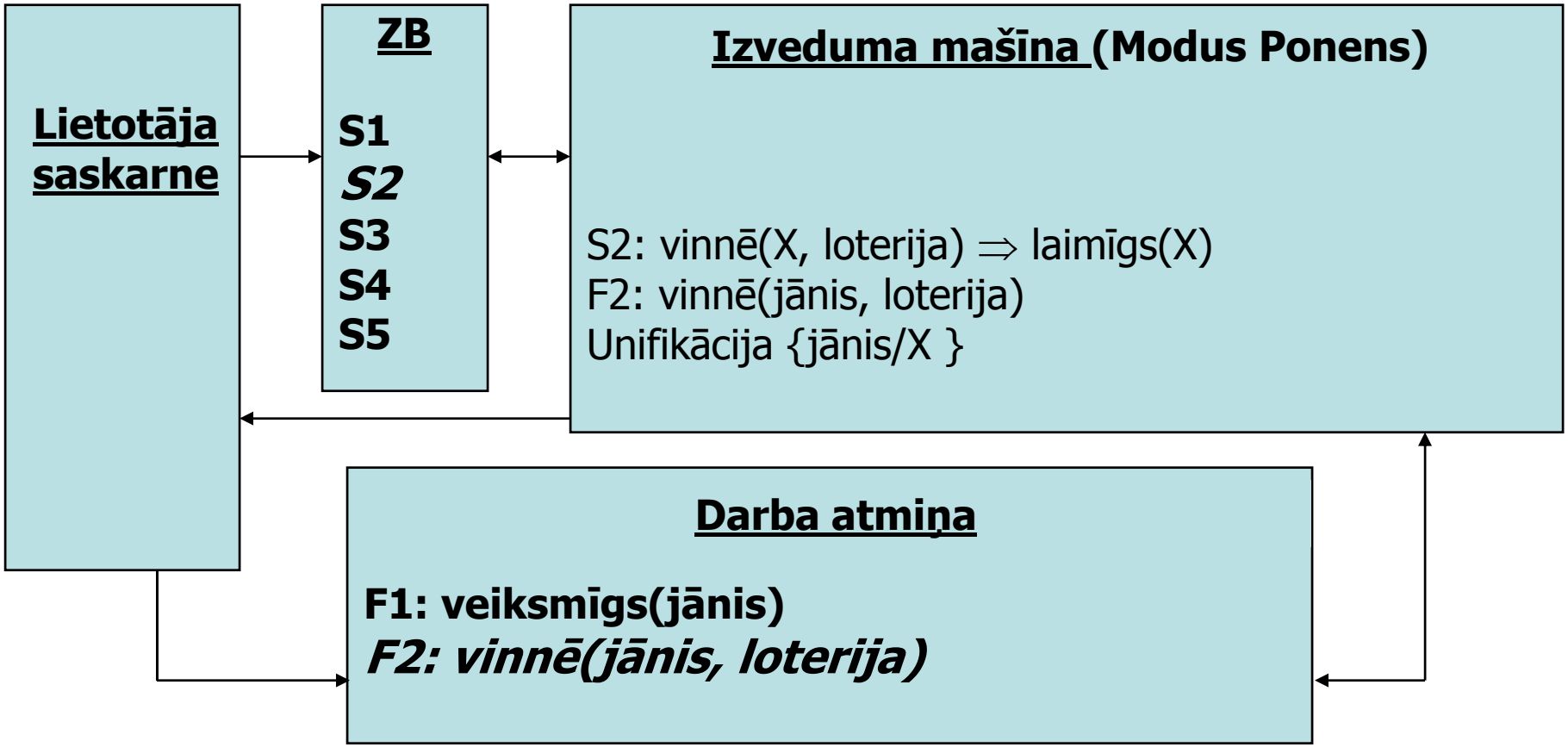
Izveduma piemērs (5)



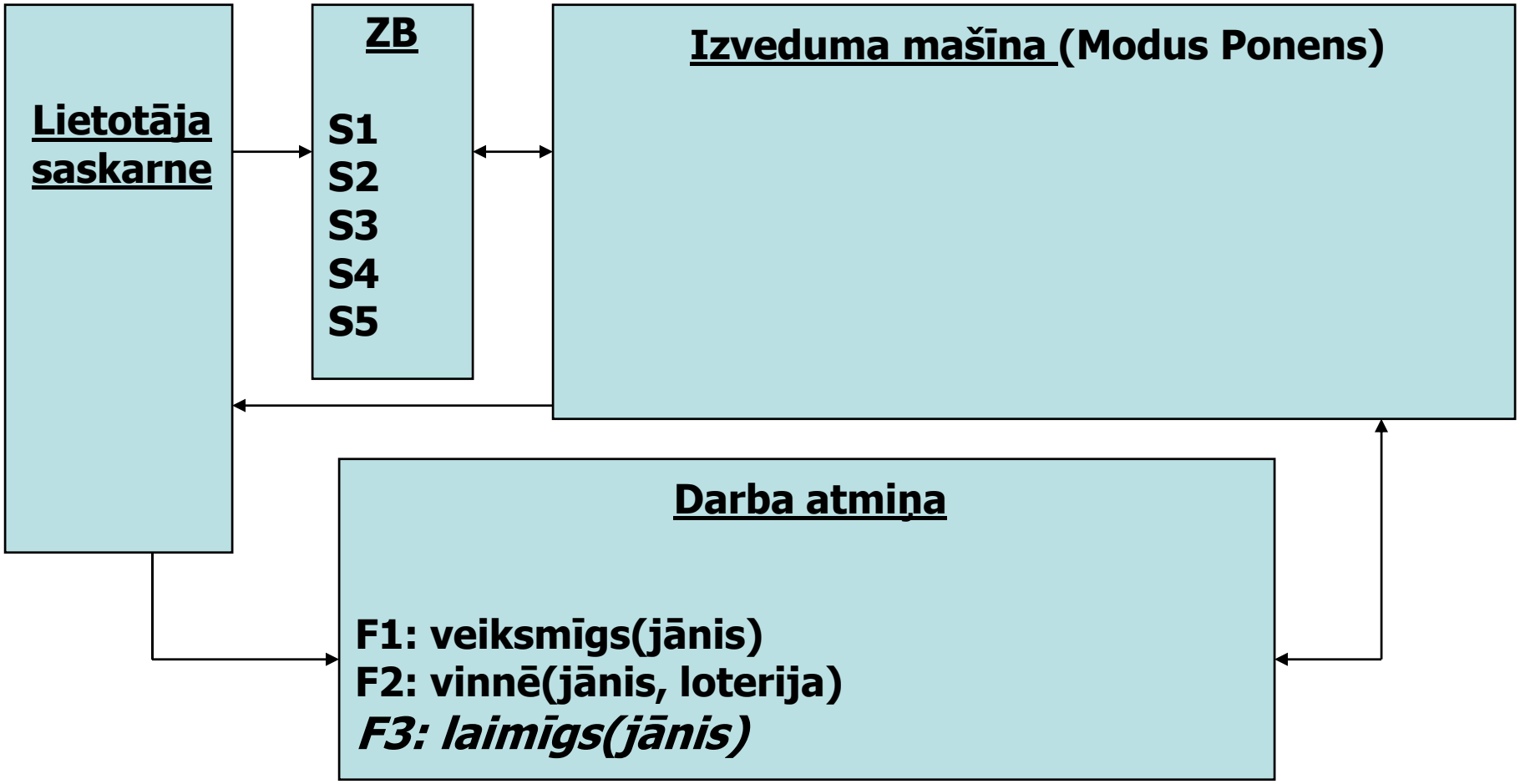
Izveduma piemērs (6)



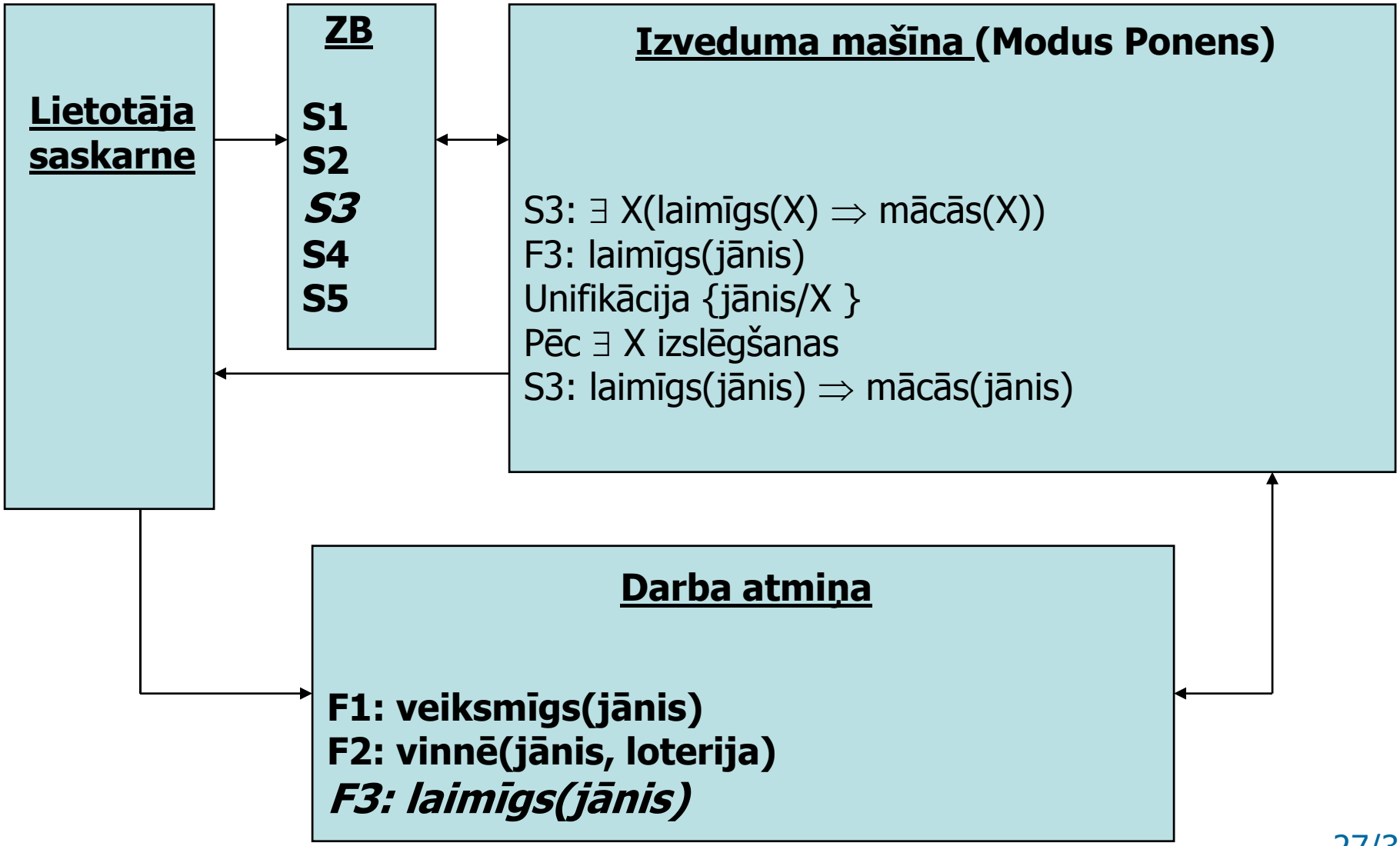
Izveduma piemērs (7)



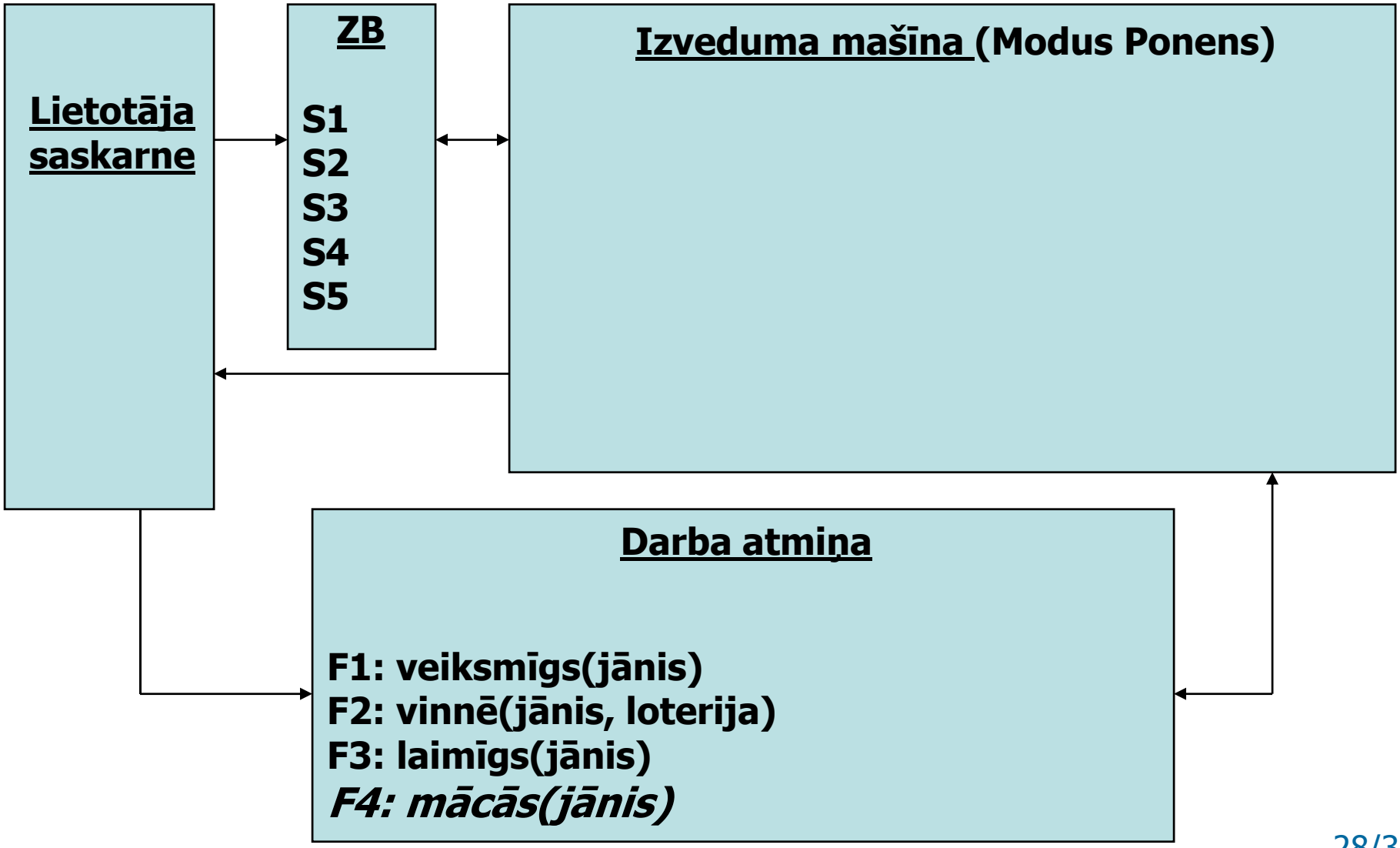
Izveduma piemērs (8)



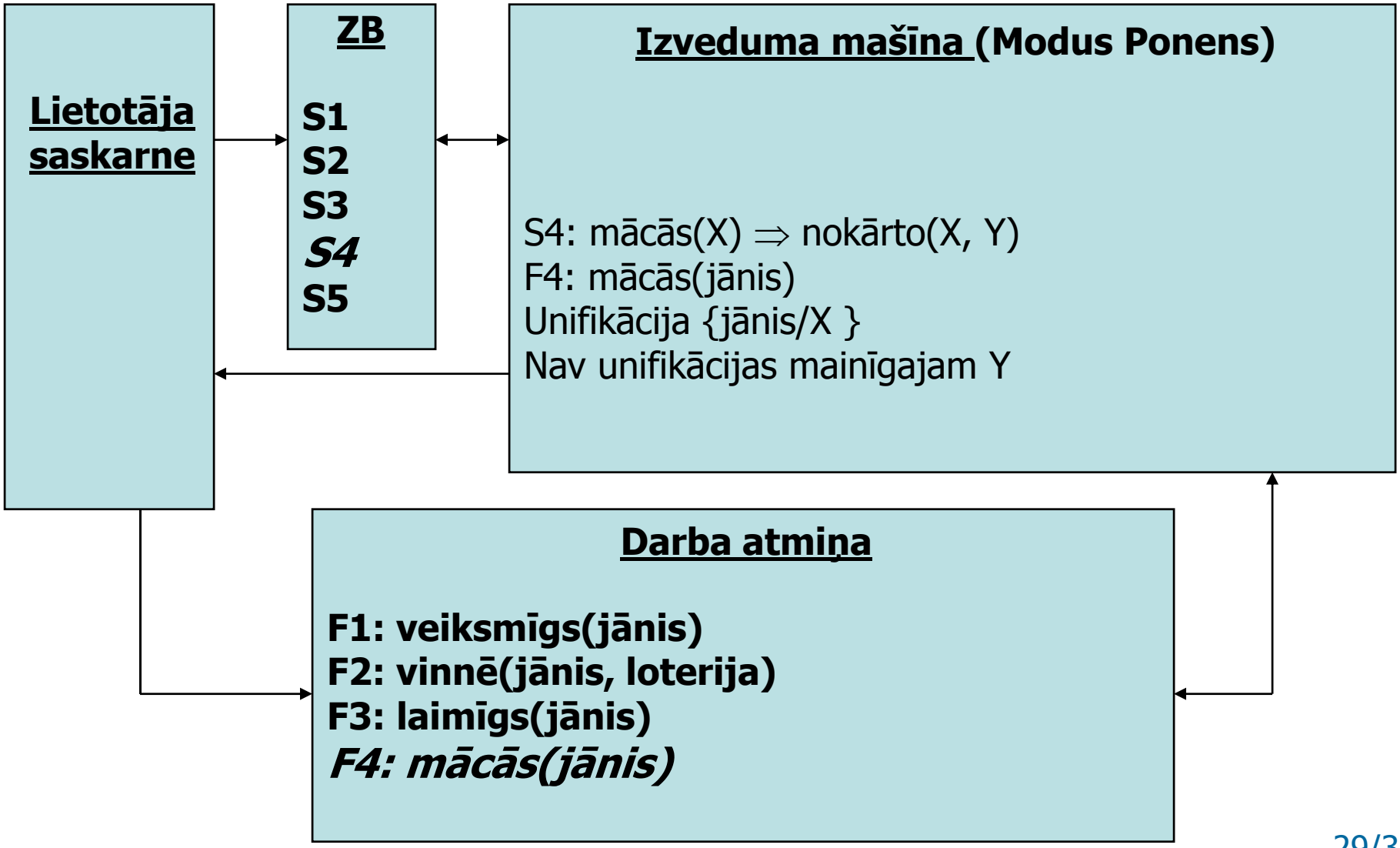
Izveduma piemērs (9)



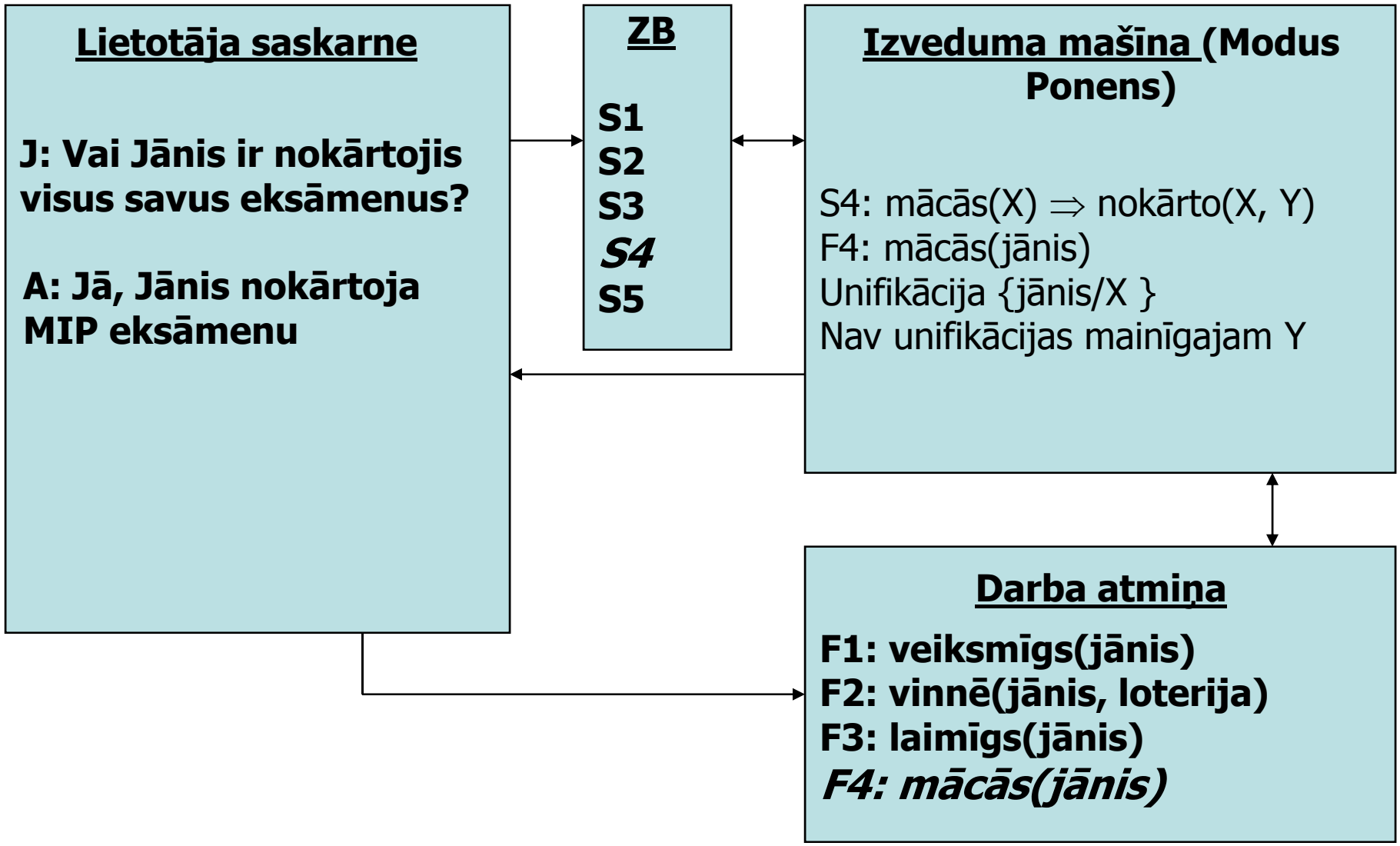
Izveduma piemērs (10)



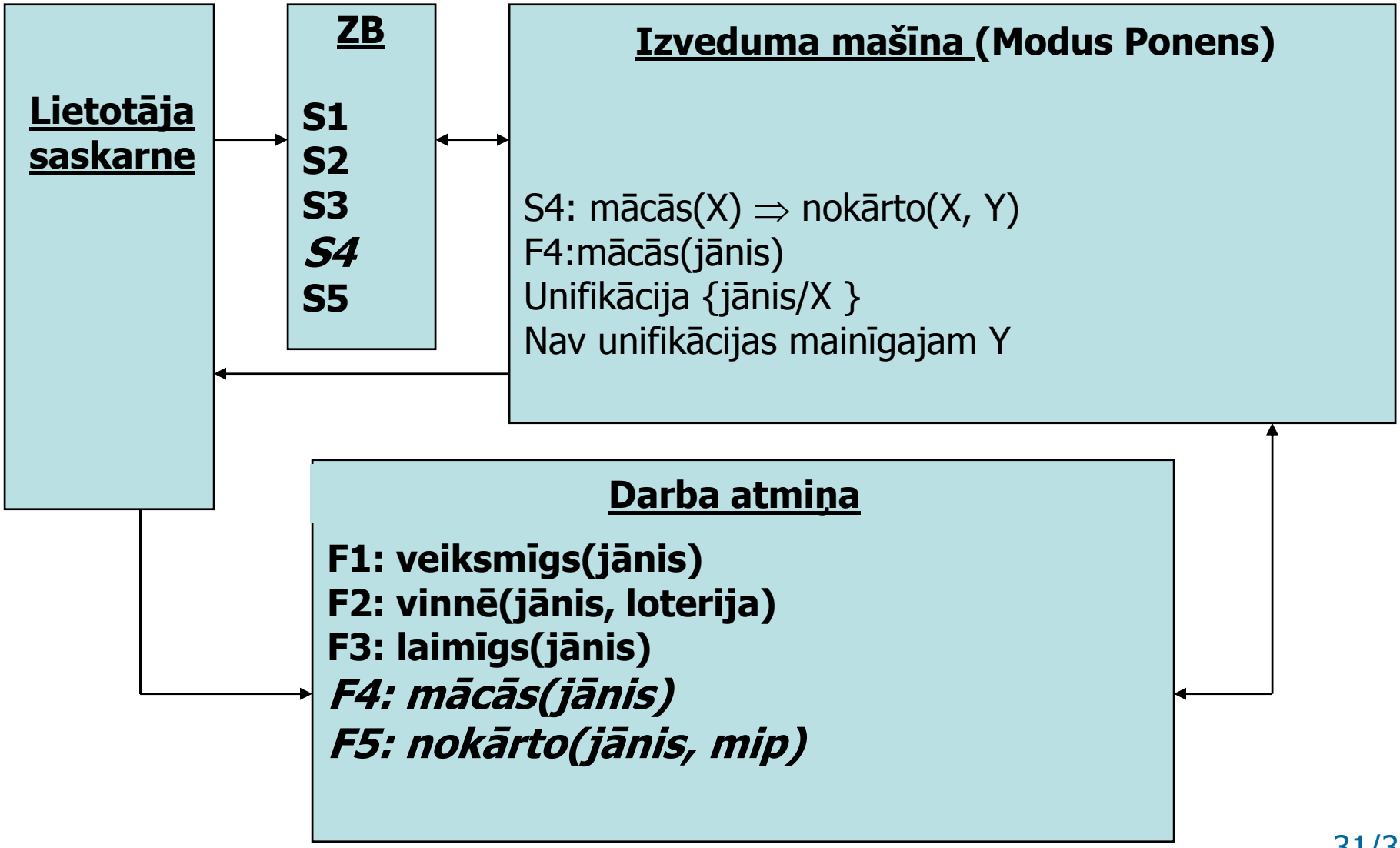
Izveduma piemērs (11)



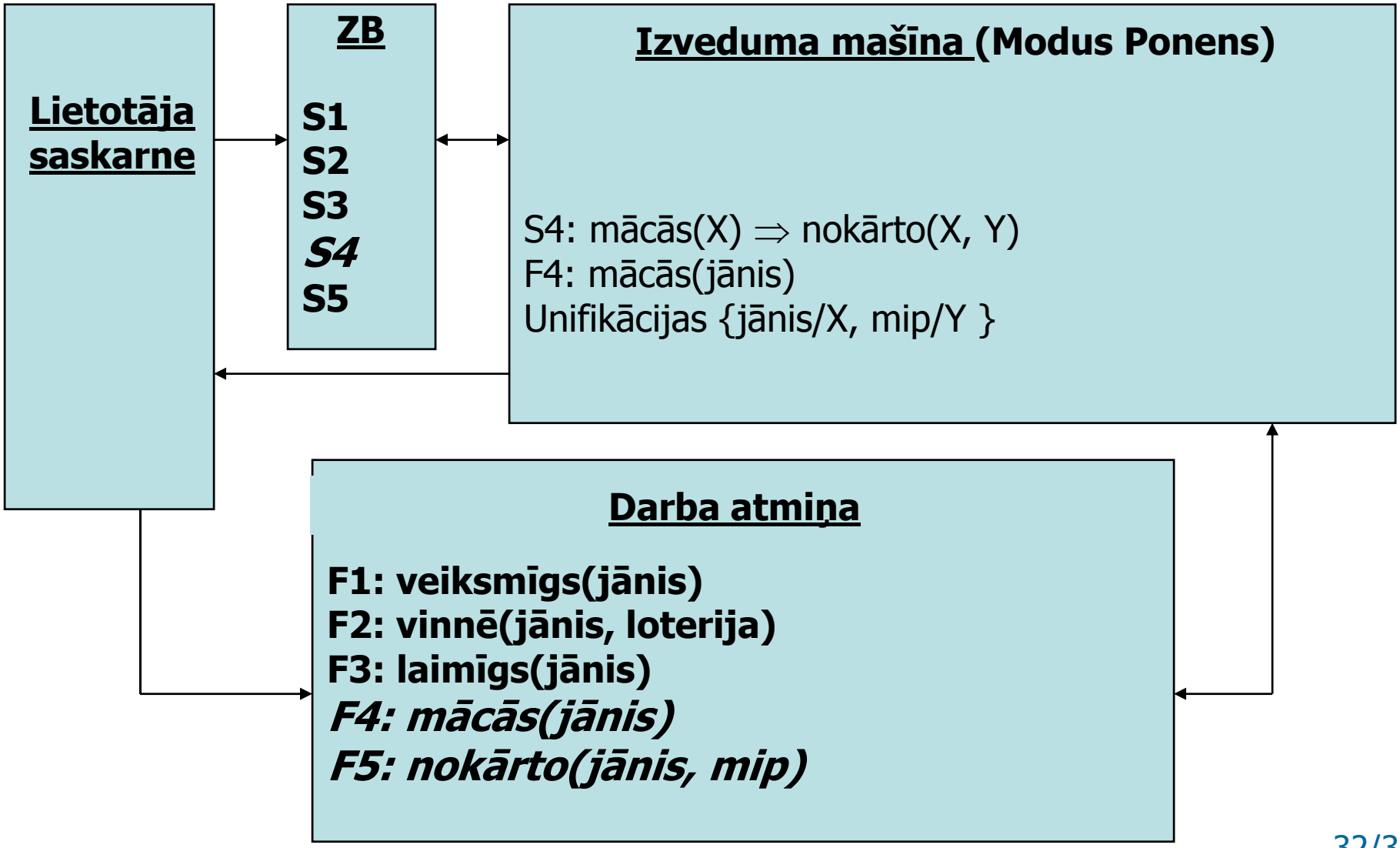
Izveduma piemērs (12)



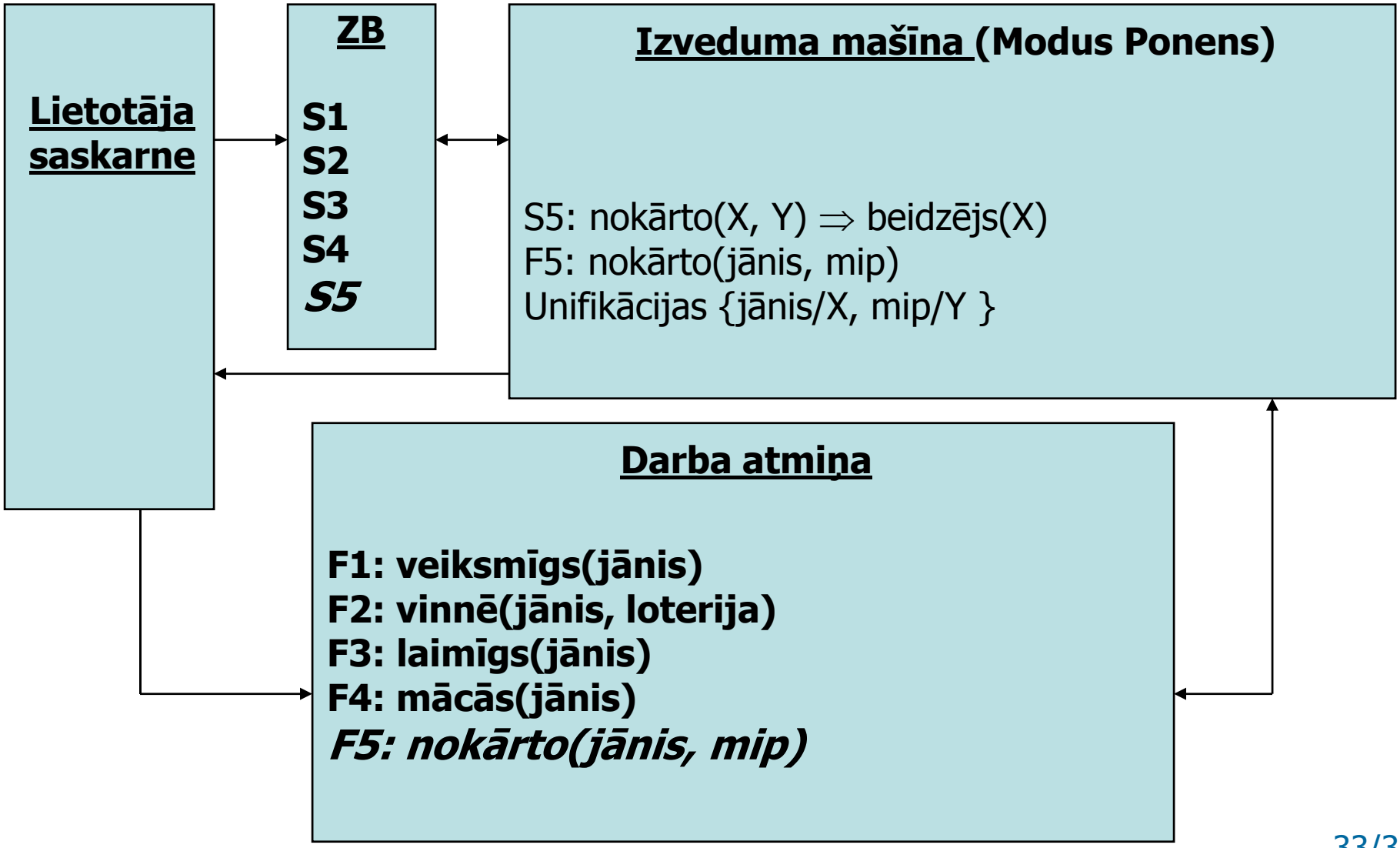
Izveduma piemērs (13)



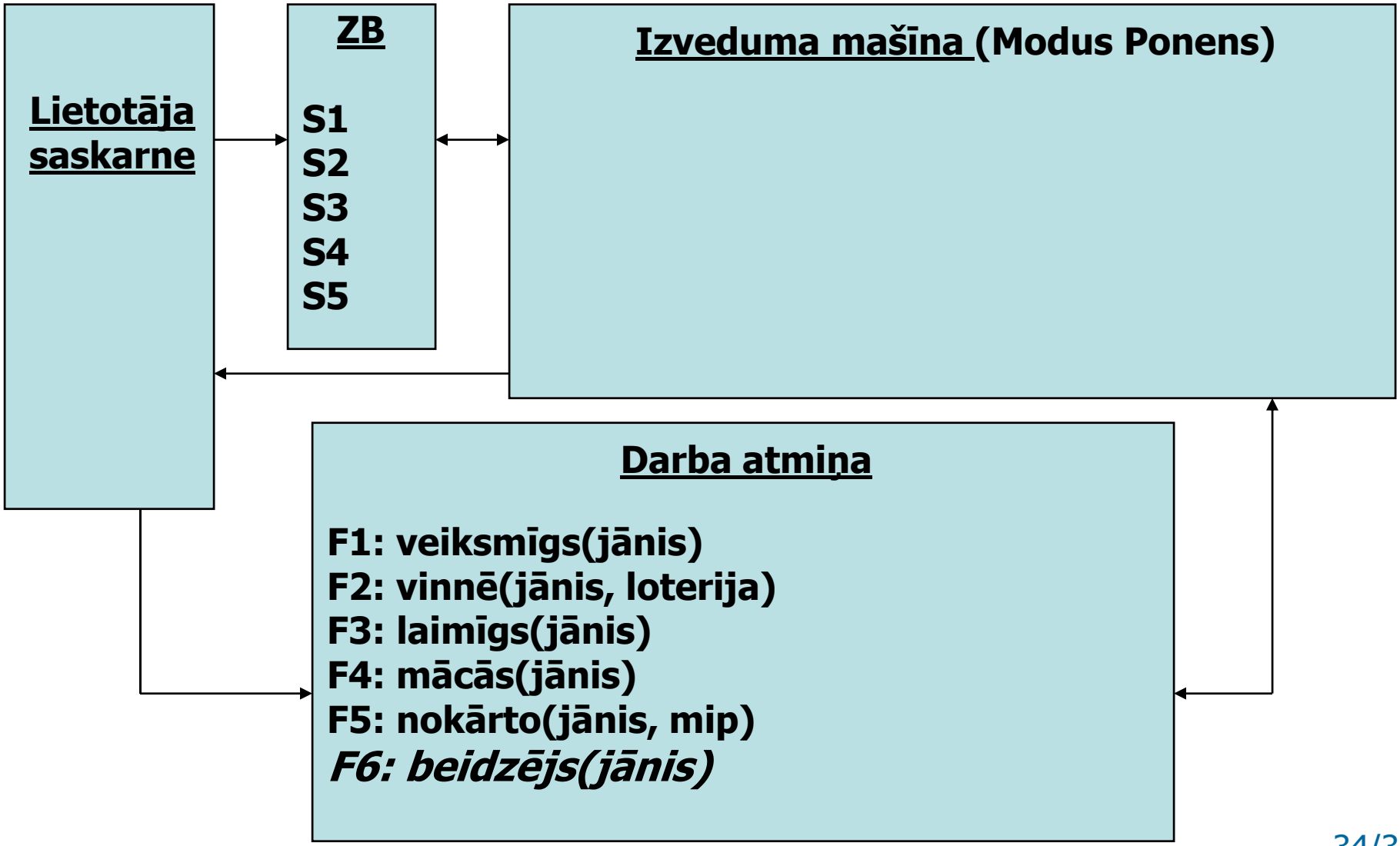
Izveduma piemērs (14)



Izveduma piemērs (15)



Izveduma piemērs (16)



Izveduma piemērs (17)