12. Algebra predikaatide rakendamine loogika valemite interpreteerimisel

12.1 Võtteid tööks termidega.

- Loogika valemite interpreteerimiseks tuleb interpreteerida alamvalemeid
- Kõige lihtsam alamvalem on atomaarne valem
- Atomaarvalem 1st järku predikaatloogikas sisaldab *terme*

- **Term**
  - *Muutujad*. Iga muutuja on term
  - *Funktsioonid*. Iga n-argumendiline avaldis $f(t_1,...,t_n)$, kus argumendid t_i on termid ja f on funktsiooni sümbol, on term.
  - Indiviidkonstante tähistavad sümbolid on 0-aarsed funktsiooni sümbolid ja seega ka termid.

- Näited termidest

  $+(x, y)$ (infiks kujul $x + y$)

  $+(x, +(y, -(z)))$ (infiks kujul $x + y - z$)

Alamtermi leidmine:

Term *S* on termi *T alamterm*, kui ta on identne termiga *T* või termis *T* sisalduva termiga. Näiteks ‘s + d’ on termi ‘k – (s + d)’ alamterm.

Näide (alamtermi leidmine)

```
subterm(T, T).                % Termid on prefiks-kujul
subterm(S, T):-
      T =..[_|Args], sub(S, Args).

sub(S, [First|Rest]):-
      subterm(S, First); sub(S,Rest).
```

Päring:

```
?- subterm(+(s,d),(-k,+(s,d))).
```

12.2. Loogikaavaldiste interpreteerimine hulgaoperatsioonide abil

Defineerime hulgaoperatsioonid operaatorkujul:

```
:-op(500,  fx, [~]).        % täiend
:-op(501, xfx, [/\]).       % ühisosa
:-op(502, xfx, [\/]).       % ühend

value(~A,C):-               % Olgu A hulgateoreetiline avaldis
   value(A,B),              %Leida avaldise A interpretatsioon
   universal_set(U),
   complement(U,B,C).
value(A\/B,C):-
   value(A,U),
   value(B,V),
   union(U,V,C).
value(A/\B,C):-
   value(A,U),
   value(B,V),
   intersection(U,V,C).
value(A,A):-    set(A).        % kas A väärtuseks on hulk?
set([]).                       % tüübi kontroll
set([_|S]):- set(S).
```

```
universal_set([a,b,c,d]).      % universaalne hulk (universumi
                               % objektid)- sõltub rakendusest
complement([],_,[]).           % hulga täiend tühihulgani on[]
complement([H|T], X,Y):-       % tail-recursion
    complement(T,X,Z),!,
    ((member(H,X),Y=Z);Y=[H|Z]).
```

```
union([], Y,Y).
union([H,X],Y,Z):-
   member(H,Y),!,
   union(X,Y,Z).                % tail-recursion
union([H|X],Y,[H,Z]):-
   union(X,Y,Z).
```

Päring:

```
?- X=[a,b], Y=[b,c], value(~((X/\~Y)\/(Y/\~X)),Z).
Z=[b,d]
```

12.3 Lausearvutuse valemite interpreteerimine

Näide: tõestada valem $c \wedge (a \vee b) \rightarrow c \wedge a$, a, b,c on lausemuutujad

Päring Prologis: `[c/\(a\/b)]?c/\a.`

Interpreteeriv programm eeldab valemite esitust operaatorkujul:

Operaatorid:

```
:- op(510, fx,  [~]).             % eitus - "¬"
:- op(520, xfy, [/\]).            % konjunktsioon
:- op(530, xfy, [\/]).            % disjunktsioon
:- op(540, xfx, [->]).            % implikatsioon
:- op(550, xfx, [?]).             % järelduvus "⊦"
```

Vastuväiteline tõestusskeem

```
Assumptions?Goal:-
    transform(Assumptions, Goal, Formula), % teisendus KNK-le
    setup(Formula, Valuation),                % lausemuutjate leidm.
    (generate(Valuation),                 %lausemuutujate väärtustamine
        value(Formula,t,Valuation),   %arvuta valemi tõeväärtus ja
        write('not valid'))               %unifitseeri väärtusega t
    ;
    write('valid').                       % Kui valemi eitus alati väär


transform([],G,~G).          % valemite listi teisendamine KNK-le
transform([H|T],G,H/\X):-   % kus tõestatav valem on eituse all.
    transform(T,G,X).        % sabarekursioon
```

```
Assumptions?Goal:-
    transform(Assumptions, Goal, Formula), % teisendus KNK-le
    setup(Formula, Valuation),             % lausemuutjate leidm.
    (generate(Valuation),                  %lausemuutujate väärtustamine
        value(Formula,t,Valuation),        %arvuta valemi tõeväärtus ja
        write('not valid'))                %unifitseeri väärtusega t
    ;
    write('valid').                        % Kui valemi eitus alati väär


setup(A,[[A|_]]):- atomic(A). % lausemuutujate leidmine
setup(~F,V):-  setup(F,V).    % eitusega valemi muutujate leidm
setup(F,V):-                  % binaarse seose muutujate leidm.
    F=..[_,A,B],              % A, B on binaarseose alamvalemid
    setup(A,X),
    setup(B,Y),
    union(X,Y,V).    % V-paaride list, kus 1. el. on muutuja nimi
                                            2. element väärtustamata
```

```
Assumptions?Goal:-
    transform(Assumptions, Goal, Formula), % teisendus KNK-le
    setup(Formula, Valuation),              % lausemuutjate leidm.
    (generate(Valuation),                  %lausemuutujate väärtustamine
        value(Formula,t,Valuation),         %arvuta valemi tõeväärtus ja
        write('not valid'))                %unifitseeri väärtusega t
    ;
    write('valid').                        % Kui valemi eitus alati väär


% ======= Lausemuutujate tõeväärtuste genererereerimine ==========
    generate([]).            %
    generate([[A,V]|T]):-    % A – lausemuutuja, V –tõeväärtus
        generate(T) ,
        (V=t; V=f).          % lausemuutujate väärtus algselt 'true'
                             % tagasivõtu korral 'false'
% ---------------------------------------------------------------
```

```
Assumptions?Goal:-
    transform(Assumptions, Goal, Formula), % teisendus KNK-le
    setup(Formula, Valuation),              % lausemuutjate leidm.
    (generate(Valuation),                  %lausemuutujate väärtustamine
        value(Formula,t,Valuation),        %arvuta valemi tõeväärtus ja
        write('not valid')) ;              % unifitseeri väärtusega t
    write('valid').                        % Kui valemi eitus alati väär

value(A,Z,V):-
    atomic(A),!,
    member([A,Z],V).                       % alamvalemite tõev. arvutus
value(~A,Z,V):-
    value(A,X,V),
    truth_table(~X,Z).
value(A/\B,Z,V):-
    value(A,X,V),
    value(B,Y,V),
    truth_table(X/\Y,Z).
value(A\/B,Z,V):-
    value(A,X,V),
    value(B,Y,V),
    truth_table(X\/Y,Z).
value(A->B,Z,V):- alue(A,X,V),
    value(B,Y,V),
    truth_table(X->Y,Z).
```

```
truth_table(t/\t, t):- !.
truth_table(_/\_, f).
truth_table(f\/f, f):- !.
truth_table(_\/_, t).
truth_table(t->f, f):- !.
truth_table(_->_, t).
truth_table(~t, f).
truth_table(_, t).
```

Temporaalloogika valemite interpreteerimine

Temporaalloogika CTL* semantika (Kripke struktuuril)

Kripke struktuur $M = \langle S, R, L \rangle$ - struktuur,

kus

S – lõplik olekute hulk

$R \subseteq S \times S$ – kõikjal määratud (vahetu-) saavutatavuse relatsioon;

$L: S \rightarrow 2^{AP}$ - märgistus (märgistab iga oleku selles olekus kehtivate Atomaarvalemitega hulgast AP).

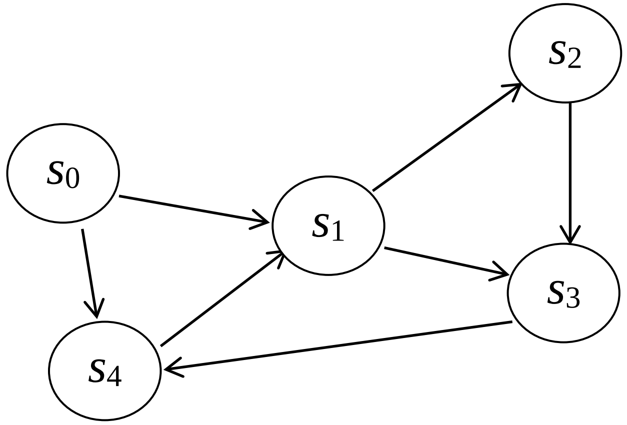
Tee struktuuril *M* on lõpmatu olekute jada $\pi = s_0, s_1, \ldots,$ kus $\forall i \geq 0, (s_i, s_{i+1}) \in R$;

π^i - π *sufiks*, mis algab tee *i*-nda olekuga s_i;

$M, s \models f$ - valem f kehtib Kripke struktuuri M olekus s;

$M, \pi \models f$ - valem f kehtib Kripke struktuuri M teel π.

Näide: Kripke struktuur



$S = \{ s_0, \ldots , s_4\}$

$R = \{\langle s_0, s_1\rangle,\langle s_0, s_4\rangle,\langle s_1, s_2\rangle,\ldots, \langle s_4, s_1\rangle\}$

$L = \{s_0 \longrightarrow \{x > 3, y = \mathrm{z}, f(\mathrm{z}) - x > 0\},$

$\ldots,$

$s_4 \longrightarrow \{x = 2, y > \mathrm{z}, f(\mathrm{z}) < x\} \}$

CTL valemite interpreteerimine

Olgu *p* atomaarvalem.

1. $M, s \models p \Leftrightarrow p \in L(s)$
2. $M, s \models \neg f \Leftrightarrow M,s \not\models f$
3. $M, s \models f_1 \vee f_2 \Leftrightarrow M,s \models f_1$ *või* $M,s \models f_2$
4. $M, s \models f_1 \wedge f_2 \Leftrightarrow M,s \models f_1$ *ja* $M,s \models f_2$
5. $M, s \models \mathbf{E}\, g \Leftrightarrow$ *alates olekust s leidub tee* π, *nii et* $M,\pi \models g$
6. $M, s \models \mathbf{A}\, g \Leftrightarrow$ *mistahes tee* π *korral olekust s* $M,\pi \models g$
7. $M,\pi \models f \Leftrightarrow$ *s on tee* π *esimene olek ja* $M,s \models f$
8. $M,\pi \models \neg g \Leftrightarrow M,\pi \not\models g$
9. $M,\pi \models g_1 \vee g_2 \Leftrightarrow M,\pi \models g_1$ *või* $M,\pi \models g_2$
10. $M,\pi \models g_1 \wedge g_2 \Leftrightarrow M,\pi \models g_1$ *ja* $M,\pi \models g_2$
11. $M,\pi \models \mathbf{X}\, g \Leftrightarrow M,\pi^1 \models g$
12. $M,\pi \models \mathbf{F}\, g \Leftrightarrow$ *leidub* $k \geq 0$, *et* $M,\pi^k \models g$
13. $M,\pi \models \mathbf{G}\, g \Leftrightarrow$ *iga* $i \geq 0$ *korral* $M,\pi^i \models g$
14. $M,\pi \models g_1\, \mathbf{U}\, g_2 \Leftrightarrow$ *leidub* $k \geq 0$, *et* $M,\pi^k \models g_2$ *ja iga* $0 \leq j < k$ *korral* $M,\pi^j \models g_1$
15. $M,\pi \models g_1\, \mathbf{R}\, g_2 \Leftrightarrow$ *iga* $j \geq 0$ *korral, kui iga* $i < j$ *korral* $M,\pi^i \not\models g_1$, *siis* $M,\pi^j \models g_2$

Samasused

$\vee$, $\neg$, **U**, **X** ja **E** kaudu saab väljendada kõiki teisi CTL* operaatoreid:

- $f \wedge g \equiv \neg(\neg f \vee \neg g)$
- $f\ \mathbf{R}\ g \equiv \neg(\neg f\ \mathbf{U}\ \neg g)$
- $\mathbf{F}\ f \equiv true\ \mathbf{U}\ f$
- $\mathbf{G}\ f \equiv \neg\mathbf{F}\ \neg f$
- $\mathbf{A}\ f \equiv \neg\mathbf{E}\ \neg f$

CTL ja LTL on CTL* alamloogikad.

CTL – hargneva ajaga loogika

LTL – lineaarse ajaga loogika

CTL: temporaalsed operaatorid **A** ja **E** on sisuliselt kvantorid antud olekust lähtuvate võimalike teede hulgal.

LTL: operaatorid kirjeldavad olekute hulki ühel teel.

CTL-s temporaalsed operaatorid **X**, **F**, **G**, **U** ja **R** võivad esineda ainult tee kvantorite järel, st CTL* teevalemeid kitsendab reegel:

- Kui f ja g on olekuvalemid, siis $\mathbf{X}\, f$, $\mathbf{F}\, f$, $\mathbf{G}\, f$, $f\, \mathbf{U}\, g$, $f\, \mathbf{R}\, g$ on teevalemid.

LTL valemitel on kuju $\mathbf{A}\, f$, kus f on teevalem ja ainukesed lubatud olekuvalemid on atomaarsed valemid:

- Kui $p \in AP$, siis p on teevalem
- Kui f ja g on teevalemid, siis on teevalemid:
  - $\neg f$, $f \vee g$, $f \wedge g$, $\mathbf{X}\, f$, $\mathbf{F}\, f$, $\mathbf{G}\, f$, $f\, \mathbf{U}\, g$, $f\, \mathbf{R}\, g$

Loogikatel CTL*, CTL ja LTL on erinev väljendusvõimsus:

Näide:

- CTL-s puudub valem, mis oleks ekvivalentne LTL valemiga **A**(**FG** *p*) - ”igal teel leidub olek, millest alates kehtib alati valem *p*”.
- LTL-s puudub valem, mis oleks ekvivalentne CTL valemiga **AG**(**EF** *p*).
- Valem **A**(**FG** *p*) ∨ **AG**(**EF** *p*) on CTL* valem, mis ei ole väljentatav ei CTL-s ega LTL-s.

CTL operaatorid:

- **AX** ja **EX**
- **AF** ja **EF**
- **AG** ja **EG**
- **AU** ja **EU**
- **AR** ja **ER**

SAMASUSED

Kõik CTL operaatorid on väljendatavad **EX**, **EG** ja **EU** kaudu:

$$\mathbf{AX}\, f \equiv \neg\, \mathbf{EX}(\neg f)$$
$$\mathbf{EF}\, f \equiv \mathbf{E}[\mathit{true}\ \mathbf{U}\ f]$$
$$\mathbf{AG}\, f \equiv \neg\, \mathbf{EF}(\neg f)$$
$$\mathbf{AF}\, f \equiv \neg\, \mathbf{EG}(\neg f)$$
$$\mathbf{A}(f\ \mathbf{U}\ g) \equiv \neg\, \mathbf{E}[\neg g\ \mathbf{U}\ (\neg f \wedge \neg g)] \wedge \neg \mathbf{EG} \neg g$$
$$\mathbf{A}(f\ \mathbf{R}\ g) \equiv \neg\, \mathbf{E}[\neg f\ \mathbf{U}\ \neg g]$$
$$\mathbf{E}(f\ \mathbf{R}\ g) \equiv \neg\, \mathbf{A}[\neg f\ \mathbf{U}\ \neg g]$$

Modaalsused CTL-s:

A – "Kõikide teede korral"

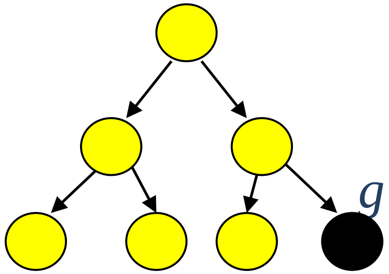
E – "Mõne tee korral"

[] – "Kõikide olekute korral antud teel"

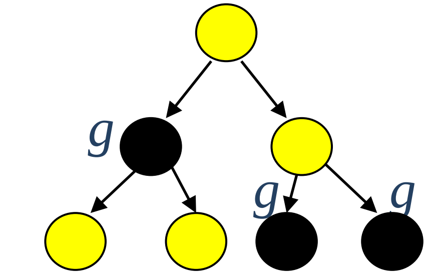
$\langle\rangle$ - "Mõne oleku korral antud teel"

Vaatame CTL fragmenti, kus kõik TL valemid on kujul ®©φ, kus

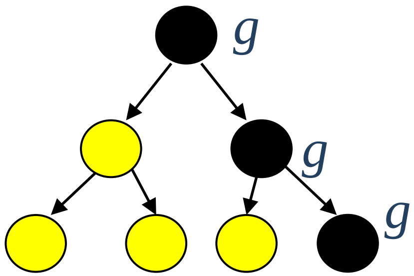
® $\in$ {A, E} ja © $\in$ { [],$\langle\rangle$}



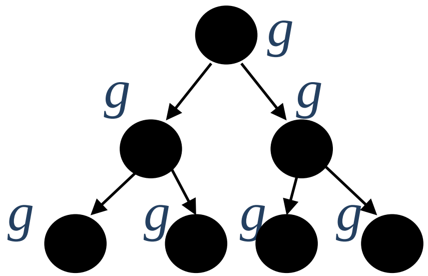
$M, s_0 \models \mathbf{EF}\ g$



$M, s_0 \models \mathbf{AF}\ g$

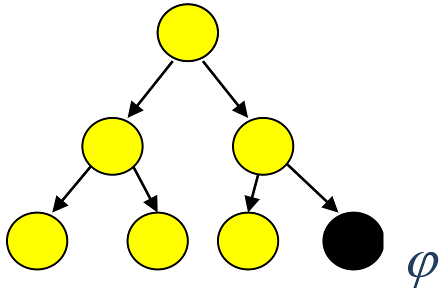


$M, s_0 \models \mathbf{EG}\ g$



$M, s_0 \models \mathbf{AG}\ g$

Tähistagu W_i indaks sammuks läbitud olekute hulka



$M, s_0 \models \mathbf{E}\langle\rangle\ \varphi$

<u>$\mathbf{E}\langle\rangle\ \varphi$:</u>

$W_{-1} := \varnothing$

$W_0 := [|\varphi|]$ % olekud, mille märgistuses sisaldub φ

$i := 0$

while $W_{i+1} \neq W_i \wedge S_0 \cap W_{i+1} = \varnothing$

do

$i := i + 1$

$W_{i+1} :=$ pre$(W_i) \cup W_i$

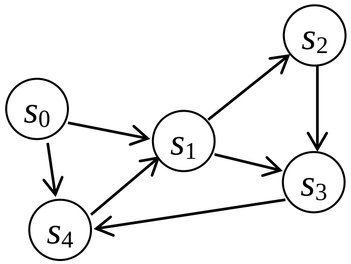
od

if $S_0 \cap W_{i+1} \neq \varnothing$ then write 'Formula $\mathbf{E}\langle\rangle\ \varphi$ is valid'

else write 'Formula $\mathbf{E}\langle\rangle\ \varphi$ is invalid'

<u>Näide:</u>

pre$(\{s_1, s_3\}) = \{s_0, s_4, s_1, s_2\}$



$\mathbf{A}\langle\rangle\ \varphi$:

$W_{-1} := \emptyset$

$W_0 := [|\varphi|]$

$i := 0$

while $W_{i+1} \neq W_i \wedge S_0 \not\subseteq W_{i+1}$

do

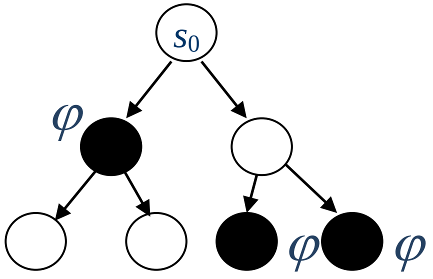
$i := i + 1$

$W_{i+1} :=$ wp$(W_i) \cup W_i$

od

if $S_0 \subseteq W_{i+1}$ then write ‘Formula **A**<> φ is valid’

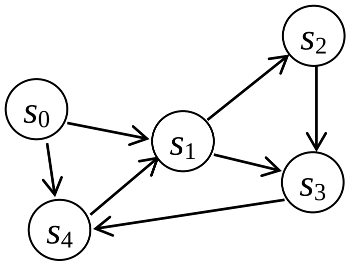
else write ‘Formula **A**<> φ is invalid’



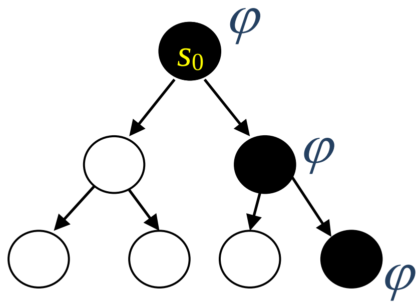
$M, s_0 \models \mathbf{A}\text{<>}\ \varphi$

Näide

wp$(\{s_1, s_3\}) = \{ s_4, s_2\}$



<u>E[] φ</u> :



$M,s_0 \models$ **E[]** φ

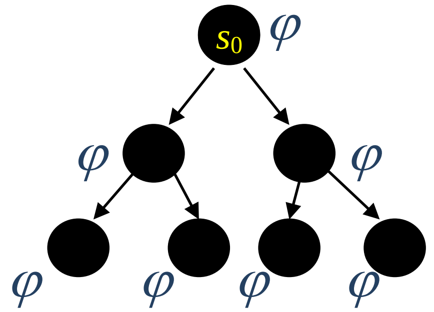
$\overline{W}_{-1} := \varnothing$
$\overline{W}_0 := \mathrm{U} \setminus [|\varphi|]$ % U – set of all states
$i := 0$
while $\overline{W}_{i+1} \neq \overline{W}_i$ % Fixed point not reached
  do
    $i := i + 1$
    $\overline{W}_{i+1} := (\mathrm{pre}(\overline{W}_i) \cap [|\varphi|]) \cup \overline{W}_i$
  od

if $S_0 \subseteq \overline{W}^*$ then write 'Formula **E[]** φ is invalid'
else write 'Formula **E[]** φ is valid'

kus $\overline{W}^*$ on algoritmi püsipunkt

A[] φ :

$\overline{W}_{-1} := \emptyset$
$\overline{W}_0 := U \setminus [|\varphi|]$ % U – set of all states
$i := 0$
while $\overline{W}_{i+1} \neq \overline{W}_i$
do
$i := i + 1$
$\overline{W}_{i+1} := (wp(\overline{W}_i) \cap [|\varphi|]) \cup \overline{W}_i$
od
if $S_0 \cap \overline{W}^* \neq \emptyset$ then write 'Formula **A[]** φ is invalid'
else write 'Formula **A[]** φ is valid'



$M, s_0 \models$ **A[]** φ

Näide: Kripke struktuur

```
rel(a,b).
rel(a,h).
rel(h,k).
rel(h,f).
rel(b,f).
rel(h,i).
rel(f,i).
rel(f,c).
rel(i,g).
rel(i,j).
rel(g,d).
rel(g,e).
state(a,[x,y,z]).
state(b,[x,y]).
state(c,[q]).
state(d,[r]).
state(e,[r]).
state(f,[x,y,q]).
state(g,[q]).
state(h,[x,y,r]).
state(i,[x,y]).
state(j,[q]).
state(k,[x,y,p]).
```

Eelkujutise leidmine

```
?- pre(rel,[f,i],Preimage).
Preimage= [h,b,f]

pre(Rel,Set, SetA):-
   assert(pre_set([])),
   pre1(Rel,Set),
   retract(pre_set(A)),list_to_set(A,SetA).

pre1(_,[]).
pre1(Rel,[El|Set]):-
   Rel_i=..[Rel,Pre_el,El],
   call(Rel_i),
   arg(1,Rel_i,Prel),
   retract(pre_set(P)),
   assert(pre_set([Prel|P])),
   fail.
pre1(Rel,[El|Set]):-
   pre1(Rel,Set).
```

Nõrgima eeltingimuse leidmine

```
wp(Rel, Set, Pre_setN):-
   pre(Rel,Set, Pre_set),
   assert(wp_set(Pre_set)),
   wp1(Pre_set,Rel,Set),
   retract(wp_set(Pre_setN)),
   write(Pre_setN),
   !.

wp1([], Rel, Set):- !.
wp1([Pre_el|Pre_set], Rel, Set):-
   Rel_i=..[Rel,Pre_el,El],
   call(Rel_i),
   arg(2,Rel_i,Post_el),
   not(member(Post_el,Set)),
   retract(wp_set(P_set)), delete(P_set,Pre_el,P_set1),
   assert(wp_set(P_set1)),
   wp1(Pre_set, Rel, Set).
wp1([Pre_el|Pre_set], Rel, Set):-
   wp1(Pre_set, Rel, Set).
```