

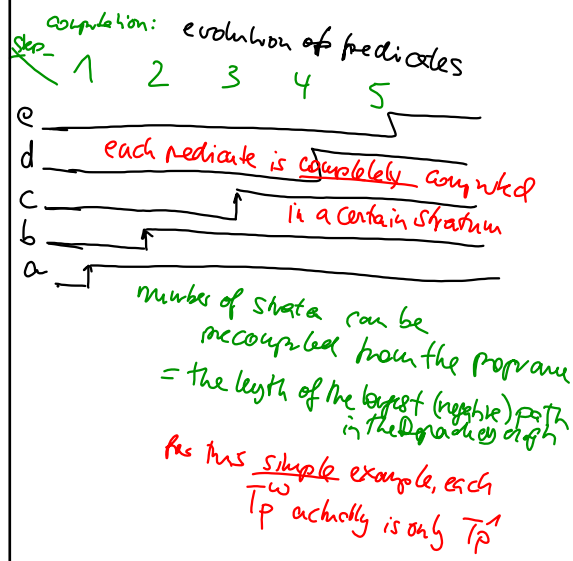
General use of rules:

Positive head atom: — possible EDB atoms,
 negative EDB atoms,
 possible IDB atoms,
 negative IDB atoms.

Jun 24-10:09

Recall Stratification: (e is EDB)

a(1)	S ₁ : a
a(3)	S ₂ : b
b(x) ← ¬a(x)	S ₃ : c
c(x) ← ¬b(x)	S ₄ : d
d(x) ← ¬c(x)	S ₅ : e
e(x) ← ¬d(x)	



Jun 24-10:55

recall positive programs :

a(1). a(2).
 $b(x) \leftarrow a(x)$.
 $c(x) \leftarrow b(x)$.
 $d(x) \leftarrow c(x)$.

Computation T_P^ω

, in this simple example = T_P^4

with a recursive rule, recall reachability (Kondratiev)
 $\Rightarrow T_P^\omega$ is necessary

Jun 24-11:03

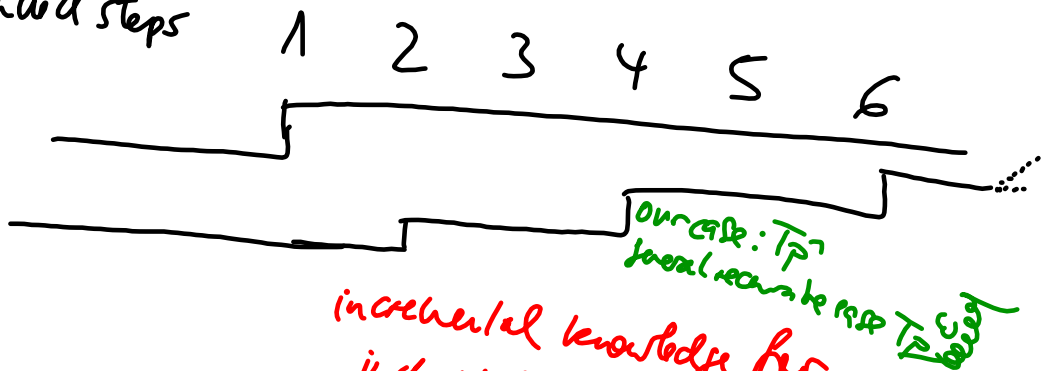
So back to well founded

$win(x) :- more(x,y), \neg win(y)$.

well founded steps

more

win



incremental knowledge for
 individual predicates

Jun 24-11:06

Considers the following extended game prog:

$w_h(x) :- \text{move}(x,y), \neg w_h(y).$

$\text{red}(x) :- w_h(x), \text{move}(x,y), w_h(y).$

~~$\text{green}(x) :- w_h(x), \neg x(\text{move}(x,y), w_h(y)).$~~

$\Rightarrow 2$ Tp-steps in each well-founded round not in De/leg!

Jun 24-11:11

Considers the reduct for that ^{extended} program

$\mathcal{R}_0 = \emptyset$; ie. $w_h = \emptyset$

$P = \{ \text{move}(a,b), \text{move}(b,k), \dots \}$ $\hat{=}$ rule $\text{move}(i,j) :- \text{true}.$
the two mbs from the previous slide \mathcal{I}_0

$P_0 = \{ \text{move}(a,b) :- \text{true}, \text{move}(b,k) :- \text{true}, w_h(a) :- \text{move}(a,b), \neg w_h(b), w_h(s) :- \text{move}(a,s), \neg w_h(\emptyset), w_h(i) :- \text{move}(b,i), \neg w_h(i), w_h(j) :- \text{move}(b,j), \neg w_h(j) \}$

Stick-deletes:

• 2nd item does not delete anything

$\Rightarrow P \mathcal{R}_0$

from the second rule:

$\text{red}(a) :- w_h(a), \text{move}(a,b), w_h(b).$

Jun 24-11:23

$$T_{P_{\alpha_0}}^1(\emptyset) = \{ \text{move}(a, f) \dots \text{move}(b, k), \dots \}$$

only the moves = EDB

$$T_{P_{\alpha_0}}^2(\emptyset) = T_{P_{\alpha_0}}^1(\emptyset) = \{ \text{move}(a, f) \dots \text{move}(b, k), \text{win}(a), \dots \text{win}(b) \}$$

$$T_{P_{\alpha_0}}^3(\emptyset) = \{ \dots \} \cup \{ \text{red}(a) \}$$

⋮

⇒ we need T_P^ω here!

=: \mathcal{K}_1

⋮

Jun 24-11:33

Consider a differently extended program:

$\text{win}(x) :- \text{move}(x, y), \neg \text{win}(y).$

$\text{lose}(y) :- \neg \text{win}(y).$

consider evaluation:

let's guess some (incomplete) intermediate stage:

$$\mathcal{K}_2^* : \{ \text{win}(a), \text{win}(b), \text{win}(i), \text{lose}(f), \text{lose}(k), \text{lose}(n), \text{lose}(j) \}$$

obvious!
(obvious) agree

Jun 24-11:43

reduct $P_{\mathcal{L}_2}$:

$P_g = \{ \text{move}(\dots) \dots \}$

- ~~win(a) :- move(a,b), win(b)~~
- win(a) :- move(a,f), win(f)
- win(b) :- move(b,k), win(k)
- ~~win(c) :- move(c,a), win(a)~~
- win(c) :- move(c,d), win(d)
- win(d) :- move(d,e), win(e)
- ~~lose(a) :- not win(a)~~
- lose(c) :- not win(f), true
- lose(f) :- not win(p), true
- lose(h) :- not win(h), true
- win(i) :- move(i,j), not win(j)
- lose(m) :- not win(m), true

(! Cover our inhibition that e is not won)
 a is not lost! because it is known to be won

$\} =: P_{\mathcal{L}_2}$

win(h) :- move(h,m), not win(m).
 win(m) :- move(m,h), not win(h).

Jun 24-11:44

evaluate $P_{\mathcal{L}_2}$:

$T_{P_{\mathcal{L}_2}}(\mathcal{L}_2)$

it will redente everything that was safely correctly denied before

$= \{ \text{move}(a,b) \dots \text{move}(\dots), \text{win}(h), \text{win}(m), \text{win}(a), \text{win}(c), \text{win}(b), \text{win}(i), \text{lose}(c), \text{lose}(h), \text{lose}(m), \text{lose}(f), \text{lose}(k), \text{lose}(l), \text{lose}(j) \dots \}$

why $T_{P_{\mathcal{L}_2}}$ again and again... no change, fix point,
 $=: \mathcal{L}_3^*$

see: \mathcal{L}_3 contains all (possible) atoms that we know that they must be true, but also some more.....

Jun 24-11:52

$P_3 = \{ \text{move}(\dots) \dots \dots$
~~$\text{win}(a) :- \text{move}(a,b), \neg \text{win}(b),$
 $\text{win}(a) :- \text{move}(a,f), \neg \text{win}(f),$
 $\text{win}(b) :- \text{move}(b,k), \neg \text{win}(k),$
 $\text{win}(c) :- \text{move}(c,a), \neg \text{win}(a),$
 $\text{win}(c) :- \text{move}(c,d), \neg \text{win}(d),$
 $\text{lose}(a) :- \neg \text{win}(a),$
 $\text{lose}(c) :- \neg \text{win}(c),$
 $\text{lose}(f) :- \neg \text{win}(f),$
 $\text{lose}(h) :- \neg \text{win}(h),$
 $\text{win}(i) :- \text{move}(i,j), \neg \text{win}(j),$
 $\text{lose}(m) :- \neg \text{win}(m),$~~
 \vdots
 $\} =: P_{\mathcal{L}_3}$
however $T_{P_{\mathcal{L}_3}}(\mathcal{L}_3)$

*a is not lost!
 because it is known to be win*

true

Jun 26-14:08

$T_{P_{\mathcal{L}_3}}(\mathcal{L}_3) = \{ \text{move}(\dots),$ are all there
 $\text{win}(a)^\checkmark, \text{win}(b)^\checkmark, \text{win}(i)^\checkmark, \dots$
 $\text{lose}(f)$
 $\} :$

much less than before!
 $= \mathcal{L}_4$

Jun 26-15:05

more abstract:

\mathcal{K}_2^* was a small set of "guaranteed" possible atoms.

Construct the reduct:

- remove all rules such that they have a negative literal that is not satisfied
- remove all negative literals that are assumed to be satisfied by negation as default
 - \rightarrow for all "unknown things" the literals are removed
 - \rightarrow remove many negative atoms without actually checking them!

this makes
denote their heads

\rightarrow result is a large set of pos. literals \mathcal{K}_2

next set (\mathcal{K}_3) will be small

next set (\mathcal{K}_4) will be large

Jun 26-15:11

Considers the whole process:

start with \emptyset

= a very small set of possible atoms

$T_{P_\emptyset}(\emptyset)$ is a large set =: \mathcal{K}_1

$T_{P_{\mathcal{K}_1}}(\emptyset)$... the P is smaller... =: \mathcal{K}_2
is small

Jun 26-15:20

Step	0	1	2	3	4	5	6
atous		all except pknj		abcd efg ghij klm lmn	abcd efg ghij klm lmn	abcd efg ghij klm lmn	.
win		abcde shikem	abi				Exercise
lose		abcd efgh ijklm n	all except abi	efgh ijklm n	efgh ijklm n	efgh ijklm n	e f j k l n
		nothing					

Jun 26-15:25