General case of rules:

positive head atom :- possible EDB atoms, negative EDB atoms, possible IDB atoms, negative IDB atoms.

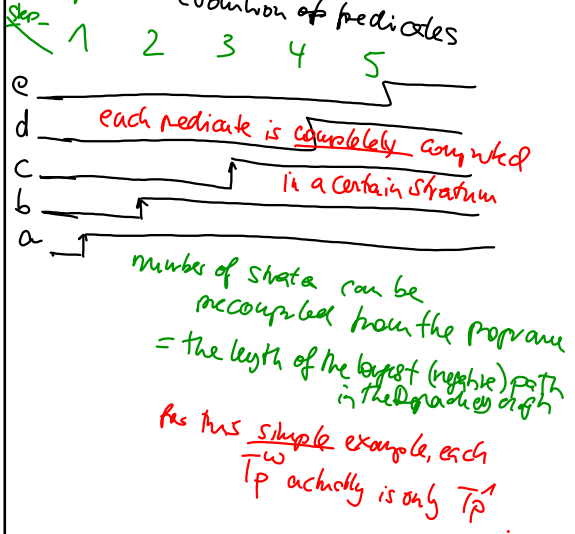
Jun 24-10:09

recall Stratification: (a is EDB)

a(1).
a(3).
b(x) ← ¬a(x)
c(x) ← ¬b(x)
d(x) ← ¬c(x)
e(x) ← ¬d(x)

S_1: a
S_2: b
S_3: c
S_4: d
S_5: e

computation: evolution of predicates



number of strata can be precomputed from the program = the length of the biggest (negative) path in the dependency graph

As this simple example, each T_P^ω actually is only $T_P^\uparrow$.

Jun 24-10:55

recall positive programs:

a(1). a(2).

b(x) ← a(x).

c(x) ← b(x).

d(x) ← c(x).

computation $T_P \uparrow \omega$, in this simple example $= T_P \uparrow 4$

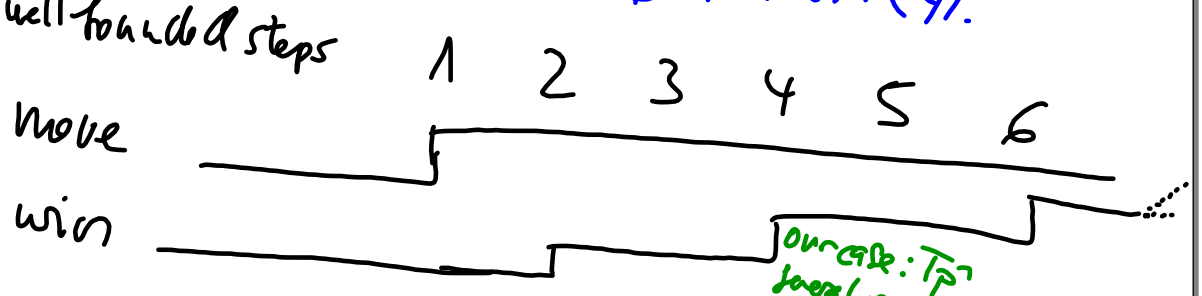
with a recursive rule, recall reachability (monadic)

$\Rightarrow T_P \uparrow \omega$ is necessary

Jun 24-11:03

So back to well founded

win(X) :- move(X,Y), ¬win(Y).



Jun 24-11:06

Consider the following extended game prg:

$win(X) :- move(X,Y), \neg win(Y).$

$red(X) :- win(X), move(X,Y), win(Y).$

~~$seen(X) :- win(X), \neg \exists Y : (move(X,Y), win(Y))$~~ not in Datalog!

$\Rightarrow$ 2 T_P-steps in each well-founded round

Jun 24-11:11

Consider the reduct for that extended program

$\mathcal{X}_0 = \emptyset$; ie. $win = \emptyset$

$P = \{ move(a,f), move(b,k), \ldots\ldots\ldots,$ $\hat{=}$ rule $move(i,j) :- true.$
the two rules from the previous slide $\}$

$P_\emptyset = \{ move(a,f) :- true,$
$move(b,k) :- true,$
$win(a) :- move(a,b), \neg win(b).$
$win(c) :- move(c,f), \neg win(f).$
$\vdots$
$win(i) :- move(b,a), \neg win(a).$
$win(i) :- move(b,k), \neg win(j).$

Side-effects:
- 2nd item does not delete anything

$\Rightarrow P_\emptyset$

Rem the second rule:

$red(a) :- win(a), move(a,b), win(b).$
$\vdots$
$\}$

Jun 24-11:23

$T^1_{P_{\mathcal{G}_0}}(\emptyset) = \{move(a,b) \dots move(b,k), \dots\}$ only the moves = EDB

$T^2_{P_{\mathcal{G}_0}}(\emptyset) = T_{P_{\mathcal{G}_0}}(\downarrow) = \{move(a,b) \dots move(b,k), win(a), \dots win(b)\}$

$T^3_{P_{\mathcal{G}_0}}(\emptyset) = \{\dots\} \cup \{red(a)\}$

$\Rightarrow$ we need $T_P{}^\omega$ here!

$=: \mathcal{H}_1$

Jun 24-11:33

Consider a differently extended program:

$win(X) :- move(X,Y), \neg win(Y).$

$lose(Y) :- \neg win(Y).$

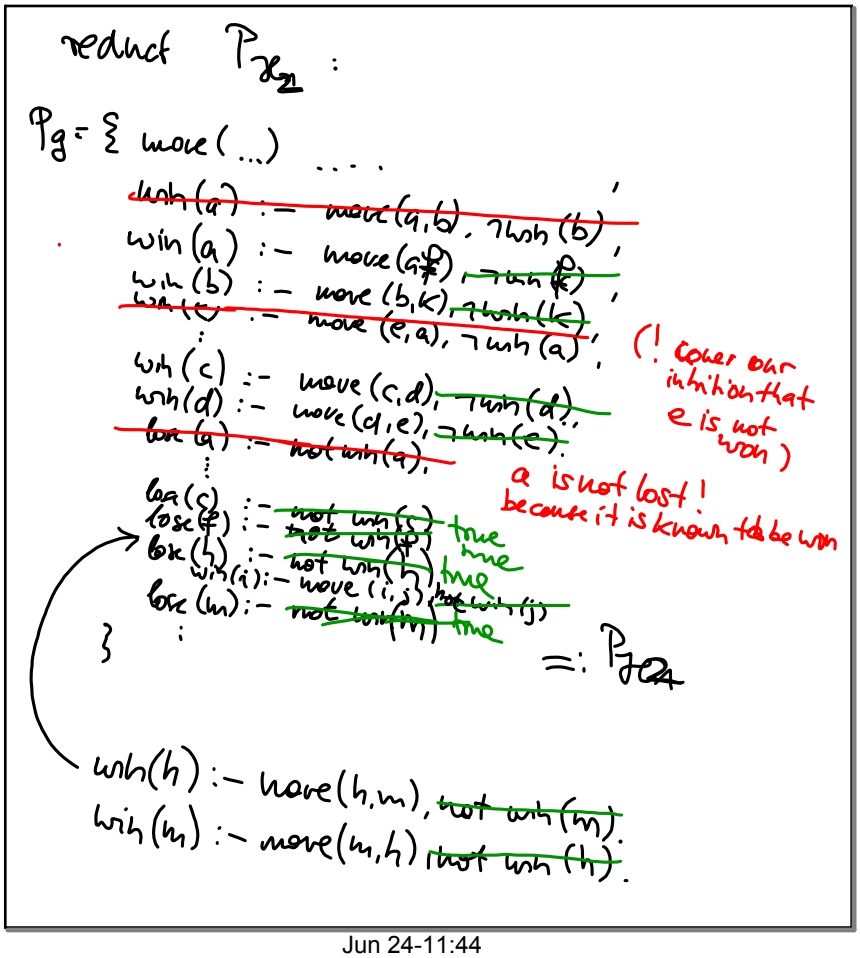
consider evaluation:

let's guess some (incomplete) intermediate stage:

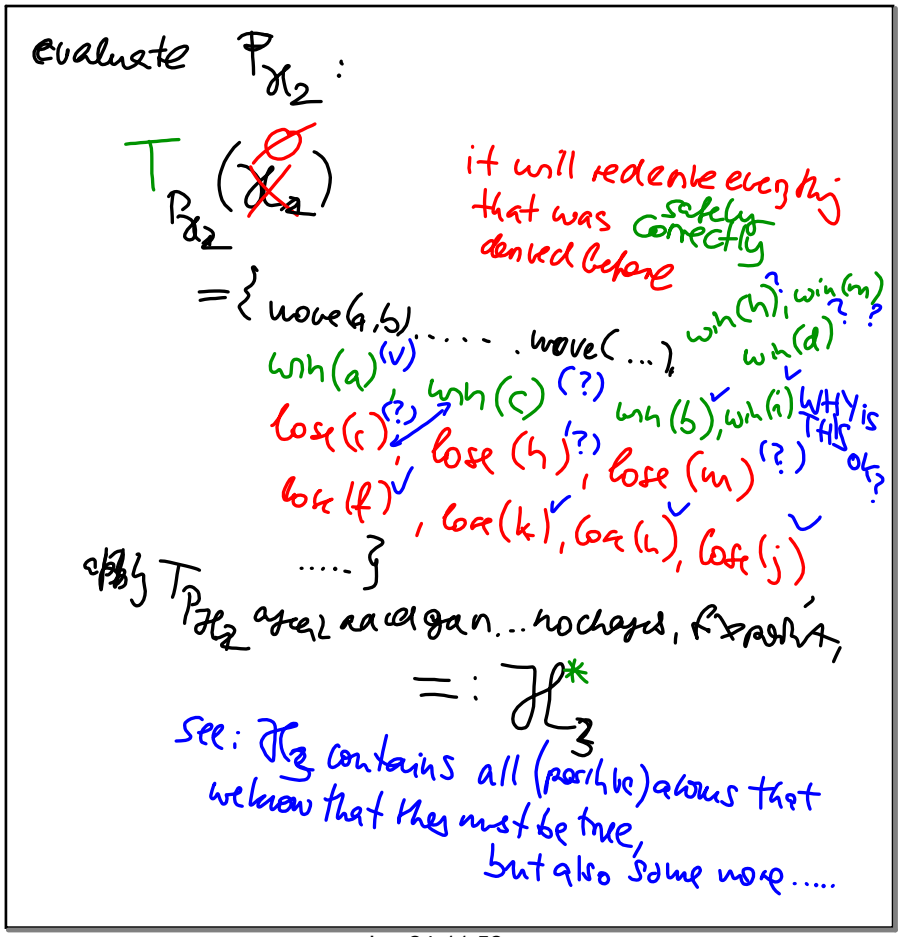
$\mathcal{H}_2^*$: $\{win(a), win(b), win(i),$ obvious!

$lose(f), lose(k), lose(h), lose(j)\}$ (obvious) agree

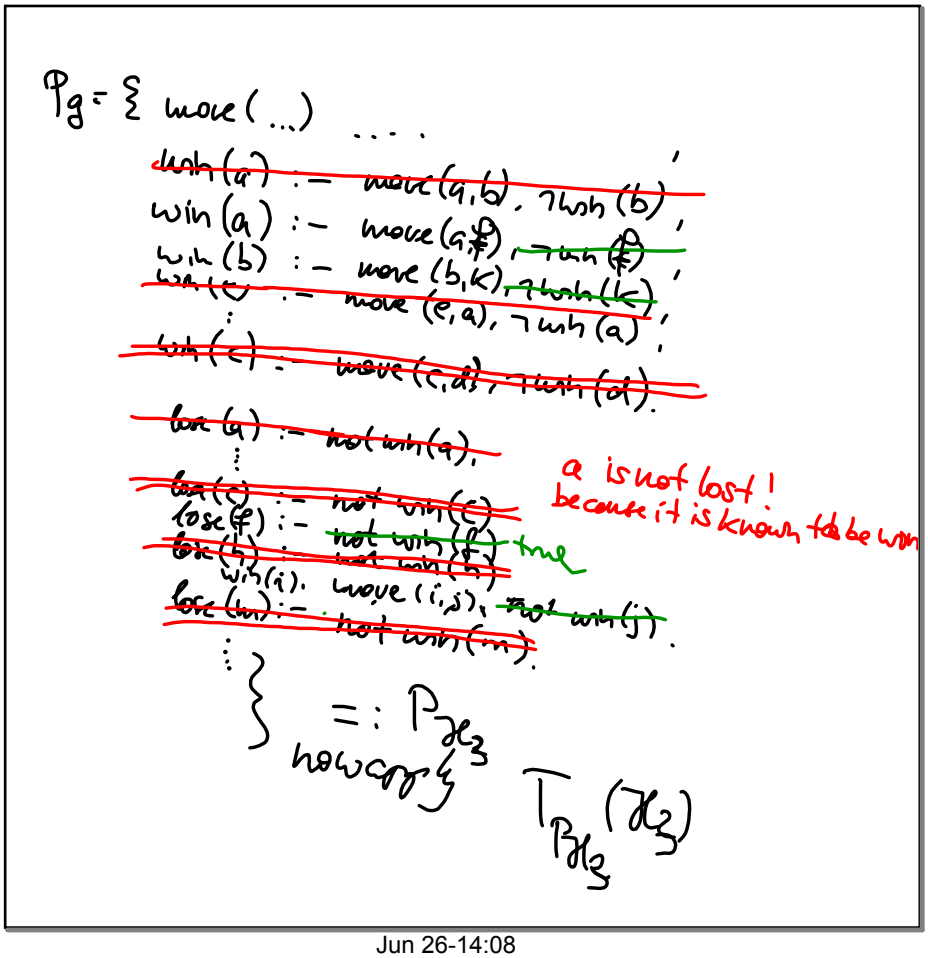
Jun 24-11:43



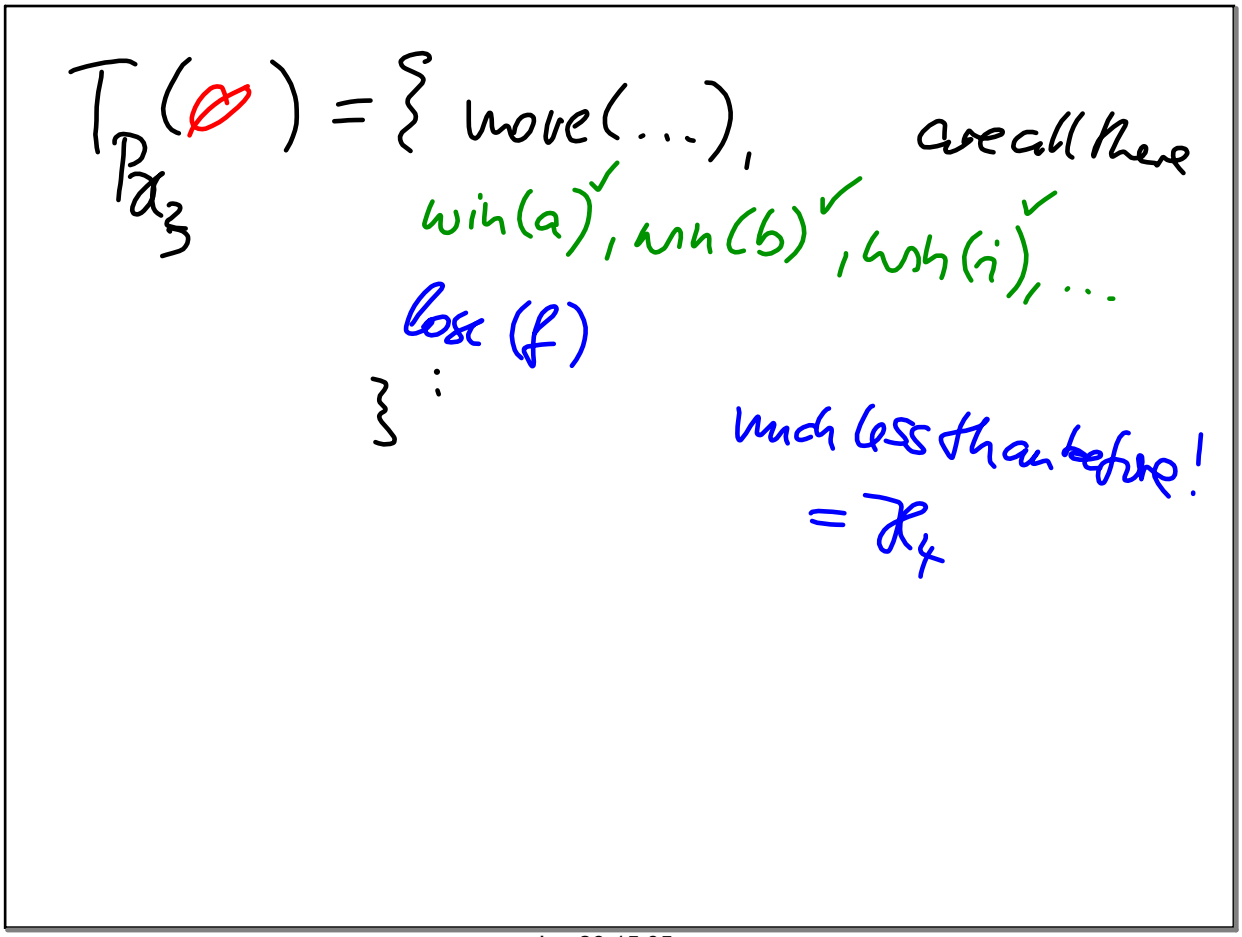
Jun 24-11:44



Jun 24-11:52



Jun 26-14:08



Jun 26-15:05

more abstract:

$\mathcal{H}_2^*$ 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
  - → for all "unknown things" the literals are removed
    - → remove many negative atoms without actually checking them!

this makes derive their head

→ result is a large set of pos. true atoms $\mathcal{H}_2$

⋮

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

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

Jun 26-15:11

Consider the whole process:

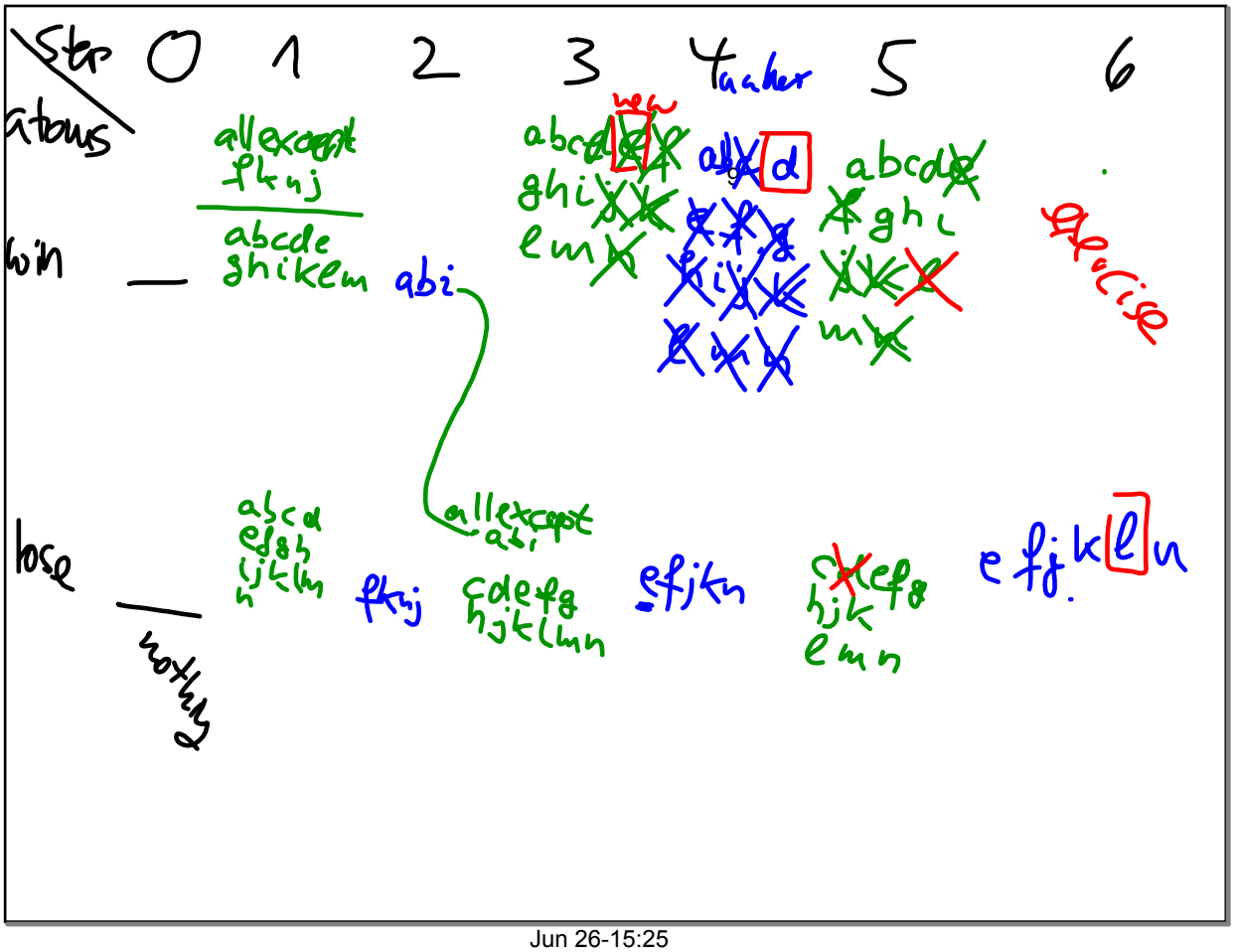
start with $\emptyset$

= a very small set of possible atoms

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

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

Jun 26-15:20


Step
Status
0
1
2
3
4
Teacher
5
6
win
lose
nothing
abi
Jun 26-15:25