generate the ground instances of rules of the program that can be useful

basis: all ground instances. Let them, as they are (for copying)

Let P_moves:={ move(a,b), move(a,f), move(b,c), move(b,), ...
all the moves facts }

collect all ground instances of the win(X) :- move(X,Y), not win(Y)
rule here that might beneeded (i.e., where the move(_,_) fact will
be derived when running T_P(emptyset)):

```
win(a) :- move(a,b), not win(b).
win(a) :- move(a,f), not win(f).
win(b) :- move(b,c), not win(c).
win(b) :- move(b,g), not win(g).
win(b) :- move(b,k), not win(k).
win(c) :- move(c,d), not win(d).
win(c) :- move(c,l), not win(l).
win(d) :- move(d,e), not win(e).
win(e) :- move(e,a), not win(a).
win(g) :- move(g,h), not win(h).
win(g) :- move(g,i), not win(i).
win(h) :- move(h,m), not win(m).
win(i) :- move(i,j), not win(j).
win(l) :- move(l,d), not win(d).
win(m) :- move(m,h), not win(h).
```

first round:

H_0 = emptyset

win(a) :- move(a,b), ~~not win(b).~~
win(a) :- move(a,f), ~~not win(f)~~.
win(b) :- move(b,c), ~~not win(c)~~.
win(b) :- move(b,g), ~~not win(g).~~
win(b) :- move(b,k), ~~not win(k).~~
win(c) :- move(c,d), ~~not win(d).~~
win(c) :- move(c,l), ~~not win(~~l).
win(d) :- move(d,e), ~~not win(e).~~
win(e) :- move(e,a), ~~not win(a).~~
win(g) :- move(g,h), ~~not win(h~~).
win(g) :- move(g,i), ~~not win(i)~~.
win(h) :- move(h,m), ~~not win(m)~~. $\mathcal{H}$
win(i) :- move(i,j), ~~not win(j).~~
win(l) :- move(l,d), ~~not win(d)~~.
win(m) :- move(m,h), ~~not win(h).~~

(P = the above + all "move"-facts)

new program P^H_0 = P_moves U {the above ground instances}

run T_P^H_0 ... \omega ... until it stops.

Here, it will stop after two T_P rounds:

T_P_H0^1(emptyset) = {moves}
T_P_H0^2(emptyset) = {moves}
U { .... the instantiated heads of these rules }
= {moves} U {win(a), win(b), win(c), win(d), win(e), win(g),
win(h), win(i), win(l), win(m) }

= T_P_H0^3(emptyset) =: H_1

... consider this result:

H_1 = {the moves} U { .... the instantiated heads of these rules }
= {moves} U {win(a), win(b), win(c), win(d), win(e), win(g),
win(h), win(i), win(l), win(m) }

note: for win(f), win(k), win(n) and win(j) there were no rules, so they have not been derived in H_1

=> we know that f,k,n,j are definitely lost positions

=> from "nothing" , we got an overestimate of the win nodes and a (safe!) underestimate of the not win/lost nodes

2nd round: H_1: win: abcdeghilm
(means: not win: fjkn)

again, build the reduct P_H_1:

first step:

delete from P_H1 all rules that contain a negative literal ¬a in the body
such that a ∈ H1,

second step:

delete all remaining negative literals in the bodies of the remaining rules.
(because those are true ... fix them intermediately)

~~win(a) :- move(a,b), not win(b).~~
win(a) :- move(a,f), ~~not win(f).~~
~~win(b) :- move(b,c), not win(c).~~
~~win(b) :- move(b,g), not win(g).~~
win(b) :- move(b,k), ~~not win(k).~~
~~win(c) :- move(c,d), not win(d).~~
~~win(c) :- move(c,l), not win(l).~~
~~win(d) :- move(d,e), not win(e).~~
~~win(e) :- move(e,a), not win(a).~~
~~win(g) :- move(g,h), not win(h).~~
~~win(g) :- move(g,i), not win(i).~~
~~win(h) :- move(h,m), not win(m).~~
win(i) :- move(i,j), ~~not win(j).~~
~~win(l) :- move(l,d), not win(d).~~
~~win(m) :- move(m,h), not win(h).~~

(P_H1 again contains all (ground) move facts)

run T_P_H1 -> omega ... finished after two rounds :

result: moves U {win(a), win(b), win(i)},
all other wins are false. => H_2
=> underestimate of true atoms

3rd round: H_2 = {the moves} U {win(a), win(b), win(i)}

as befor, now build the reduct P_H_2:

first step:

delete from P_H2 all rules that contain a negative literal ¬a in the body
such that a ∈ H2,

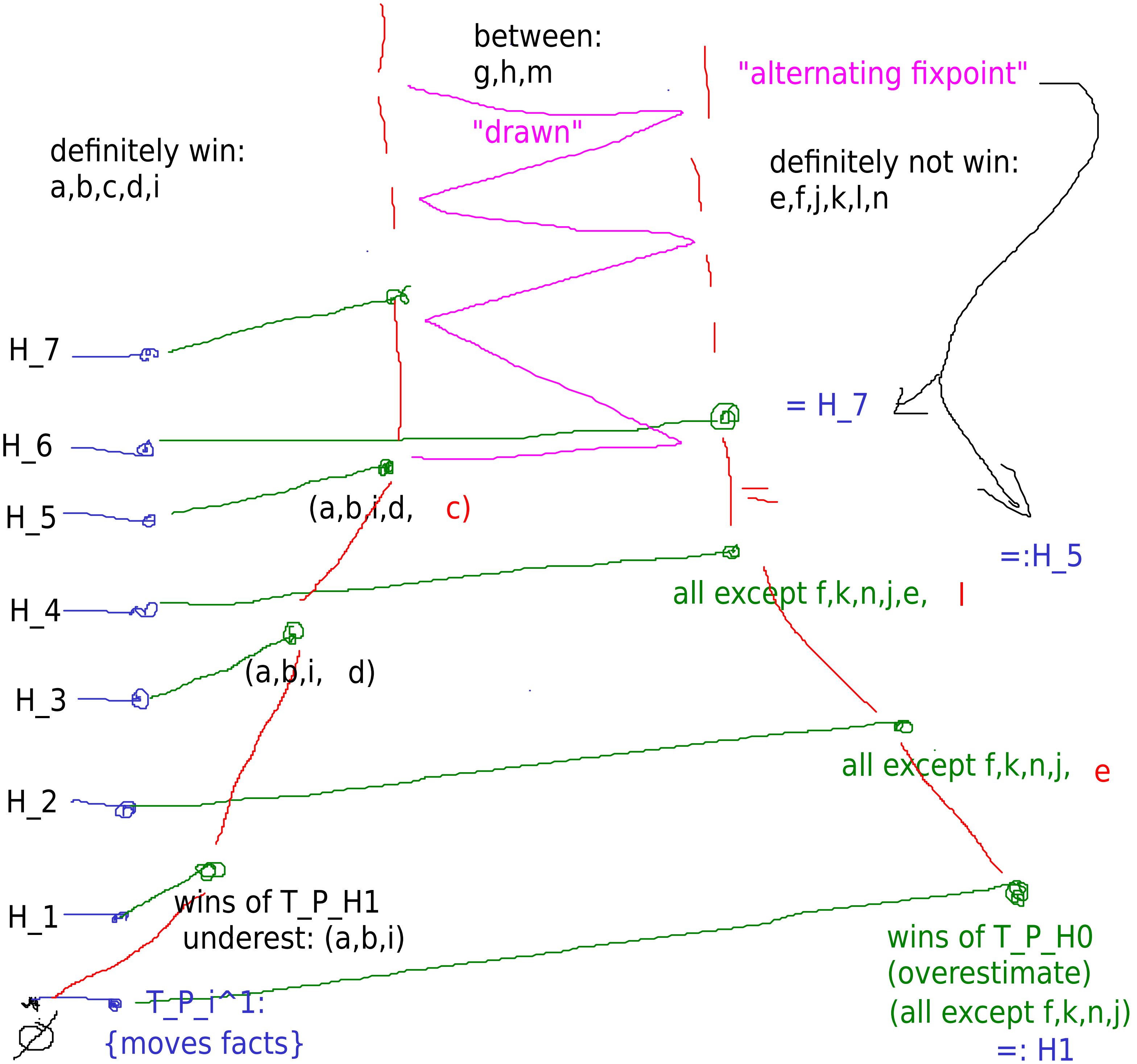
second step:

delete all remaining negative literals in the bodies of the remaining rules.
(because those are true ... fix them intermediately)

~~win(a) :- move(a,b), not win(b).~~
win(a) :- move(a,f), ~~not win(f).~~
win(b) :- move(b,c), ~~not win(c).~~
win(b) :- move(b,g), ~~not win(g).~~
win(b) :- move(b,k), ~~not win(k).~~
win(c) :- move(c,d), ~~not win(d).~~
win(c) :- move(c,l), ~~not win(l).~~
win(d) :- move(d,e), ~~not win(e).~~
~~win(e) :- move(e,a), not win(a).~~
win(g) :- move(g,h), ~~not win(h).~~
~~win(g) :- move(g,i), not win(i).~~
win(h) :- move(h,m), ~~not win(m).~~
win(i) :- move(i,j), ~~not win(j).~~
win(l) :- move(l,d), ~~not win(d).~~
win(m) :- move(m,h), ~~not win(h).~~

run T_P_H2 -> omega ... finished again after two rounds:

result: moves U {win: a,b,c,d,g,h,i,l,m}
... what is missing: not win: e, f, j, k, n
=> overestimate of win, but some (more) are known to be definitively lost



between:
g,h,m

"alternating fixpoint"

"drawn"

definitely win:
a,b,c,d,i

definitely not win:
e,f,j,k,l,n

H_7

= H_7

H_6

H_5

(a,b,i,d, c)

=:H_5

H_4

all except f,k,n,j,e, l

H_3

(a,b,i, d)

H_2

all except f,k,n,j, e

H_1

wins of T_P_H1
underest: (a,b,i)

wins of T_P_H0
(overestimate)
(all except f,k,n,j)
=: H1

∅

~~T_P_i^1:~~
{moves facts}

P_H_i always contains P_moves and the reduct wrt H_i

H_6: win: a,b,i,d,c

~~win(a) :- move(a,b), not win(b).~~
win(a) :- move(a,f), ~~not win(f).~~
~~win(b) :- move(b,c), not win(c).~~
win(b) :- move(b,g), ~~not win(g).~~
win(b) :- move(b,k), ~~not win(k).~~
~~win(c) :- move(c,d), not win(d).~~
win(c) :- move(c,l), ~~not win(l).~~
win(d) :- move(d,e), ~~not win(e).~~
~~win(e) :- move(e,a), not win(a).~~
win(g) :- move(g,h), ~~not win(h).~~
~~win(g) :- move(g,i), not win(i).~~
win(h) :- move(h,m), ~~not win(m).~~
win(i) :- move(i,j), ~~not win(j).~~
~~win(l) :- move(l,d), not win(d).~~
win(m) :- move(m,h), ~~not win(h).~~

T_P_H5^1 (emptyset): move facts
T_P_H5^2(emptyset) ->   win: a,b,c,d,g,h,i,m   =: H7 = H5
   = all except e,f,j,k,l,n