HOMEWORK 4. SOLUTIONS.

Qn 1. (Empirical Homework): See if CHAFF solver changed the dynamics between Blakbox and GP-CSP since 2000. Basically, compare GP-CSP and Blackbox (with CHAFF solver) on the problems listed in the GP-CSP paper as the ones that GP-CSP does better over Blackbox (e.g. rocket-ext-b and hsp-bw-4). See if the relative performance changes. Look at GP-CSP AIJ paper to find out what were the best settings for it, and use those settings. For Blackbox, use CHAFF solver. In both cases, use just the single solver (no shifting between solvers).

NA

Qn 2. Consider the planning graph from Qn III.a (the PG with static interferences alone) in Homework 3. Convert it into CSP encoding. Compare it to the encoding you got in Qn V of Homework 3 (which is a CSP encoding with propagated mutexes). Show that doing 3-consistency enforcement will bring in the additional constraints present in the second encoding (it is enough if you show the derivation of one of the additional constraints)

CSP Encoding:

P-2 : {#, NooP-P2}, Q-2:{#,NooP-Q2,A1-2}, R-2:{#,Noop-R2},
~Q-2:{#,Noop-~Q2,A2-2}, W-2:{#,NooP-W2,A2-2}
P-1: {#, NooP-P1}, Q-1:{#,A1-1}, R-1:{#,Noop-R1},
~Q-1:{#,A2-1}, W-1:{#,A2-1}

P-0:{#,T}, R-0:{#,T}
Goals: Q-2!=#, W-2!=#

Activation Constraints:
P-2=NooP-P2 => P-1!=#, Q-2=NooP-Q2 => Q-1!=#,Q-2=A1-2 => P-1!=#
R-2=NooP-R2 => R-1!=#, ~Q-2=NooP-~Q2 => ~Q-1 !=#, ~Q-2=A2-2 => R-1!=#
W-2=NooP-W2 => W-1 !=#, W-2=A2-2 => R-1!=#

P-1=NooP-P1 => P-0!=#, Q-1=A1-1 => P-0!=# R-1=NooP-R1 => R-0!=#,
~Q-1=A2-1 => R-0!=#, W-2=A2-1=> R-0!=#

Mutex Constraints:
Q-2=A1-2 => ~Q-2!=A2-2, Q-2=A1-2 => W-2!=A2-2,
Q-2=A1-2 => ~Q-2!=NooP-~Q2, Q-2=Noop-Q2 => ~Q-2!=A2-2,
Q-2=Noop-Q2 => W-2!=A2-2, Q-2=Noop-Q2 => ~Q-2!=NooP-~Q2,
~Q-2=A2-2 => Q-2!=A1-2, ~Q-2=A2-2 => Q-2!=NooP-Q2,
~Q-2=NooP-~Q2 => Q-2!=A1-2, ~Q-2=NooP-~Q2 => Q-2!=NooP-Q2
W-2=A2-2 => Q-2!=A1-2,W-2=A2-2 => Q-2!=NooP-Q2,

Q-1=A1-1 => ~Q-1!=A2-1, Q-1=A1-1 => W-1!=A2-1,
~Q-1=A2-1 => Q-1!=A1-1, W-1=A2-1 => Q-1!=A1-1,

Solution:

| | | | | |
|---|---|---|---|---|
| Q-2=A1-2, | P-2=NooP-P2, | R-2=NooP-R2, | ~Q-2=#, | W-2=# |
| Q-1=#, | P-1=NooP-P1, | R-1=NooP-R1, | ~Q-1=A2-1 | W-1=A2-1 |
| P-0=T, | R-O=T | | | |

NOTE: For this particular problem, we have the same kind of CSP encoding that III.b. There are only additional propositional mutexes in the graph for III.b, which can not be encoded in the csp formulation. So, nothing else is required for this problem.

Qn 3. Consider our old problem (Problem III from previous homework) again
There are two actions: A1 and A2
A1: prec: p eff: q
A2 prec: r eff: ~q,w
We start with init state where p and r are true.
**and our goals are q and w.
Common things for state-based proofs:

Propositions corresponding to initial conditions and goals are true at their respective levels (2 –STEP SAT encoding):
P-0&R-0&Q-2&W-2

At least one of the actions at each level will occur:
A1-1 V A2-1
A1-2 V A2-2

Actions imply their preconditions and effects:
A1-2 => P-1 & Q-2, A2-2=> R-1&~Q-2&W-2,
A1-1 => P-0 & Q-1 A2-1=> R-0&~Q-1&W-1

**3.a.** Write the 2-step SAT encoding based on progression proof (write all the constraints)
A proposition P at j remains true if no action occurring at j+1 deletes P.

P-0 & A1-1 => P-1, P-0 & A2-1 => P-1
R-0 & A1-1 => R-1, R-0 & A2-1 => R-1
P-1 & A1-2 => P-2, P-1& A2-2 => P-2
R-1 & A1-2 => R-2, R-1 & A2-2 => R-2
W-1 & A1-2 => W-2 W-1 & A2-2 => W-2

No more than one action occurs at each step.
~A1-1 V ~A2-1
~A1-2 V ~A2-2

**3.b.** Write the 2-step SAT encoding based on the regression proof.
A proposition P changes values between j and j+1 only if an action occurs that makes it so:

~Q-0 & Q-1 => A1-1, ~Q-1&Q-2=>A1-2
~(~Q-0) & ~Q-1 => A2-1, ~(~Q-1) & ~Q-2 => A2-2,
~W-0&W-1 => A2-1, ~W-1&W-2 => A2-2,

No pair of interacting actions must occur together:
~A1-1 V ~A2-1
~A1-2 V ~A2-2

**3.c.** Write the 2-step SAT encoding based on the causal proof.

Each step is mapped to exactly one action:
S1=A1 V S1=A2; ~(S1=A1 & S1=A2)
S2=A1 V S2=A2; ~(S2=A1 & S2=A2)

A step inherits the needs, adds and deletes of the action it is mapped to:

S1=A1 => Adds(S1,Q) & Needs(S1,P)

S1=A2 => Adds(S1,W) & Needs(S1,R) & Deletes(S1,Q)
S2=A1 => Adds(S2,Q) & Needs(S2,P)
S2=A2 => Adds(S2,W) & Needs(S2,R) & Deletes(S2,Q)

A step gets needs, adds and deletes only through mapped actions:

Adds(S1,Q) => S1=A1
Needs(S1,P) => S1=A1
Adds(S1,W) => S1=A2
Needs(S1,R) => S1=A2
Deletes(S1,Q) => S1=A2

Adds(S2,Q) => S2=A1
Needs(S2,P) => S2=A1
Adds(S2,W) => S2=A2
Needs(S2,R) => S2=A2
Deletes(S2,Q) => S2=A2

Every need is established by some step:
Needs(S0,P), Needs(S0,R)
Needs(S1,P) => Estab(S0,P,S1) , Needs(S1,R) => Estab(S0,R,S1)
Needs(S2,P) => Estab(S0,P,S2) V Estab(S1,P,S2)
Needs(S2,R) => Estab(S0,R,S2) V Estab(S1,R,S2)
Needs(S8,Q) => Estab(S1,Q,S8) V Estab(S2,Q,S8)
Needs(S8,W) => Estab(S1,W,S8) V Estab(S2,W,S8)

Establishment with causal links:
Estab(S0,P,S1) => Link(S0,P,S1)
Estab(S0,R,S1) => Link(S0,R,S1)
Estab(S0,P,S2) => Link(S0,P,S2)
Estab(S1,P,S2) => Link(S1,P,S2)
Estab(S0,R,S2) => Link(S0,R,S2)
Estab(S1,R,S2) => Link(S1,R,S2)
Estab(S1,Q,S8) => Link(S1,Q,S8)
Estab(S2,Q,S8) => Link(S2,Q,S8)
Estab(S1,W,S8) => Link(S1,W,S8)
Estab(S2,W,S8) => Link(S2,W,S8)

Links implies addition & precedence
Link(S0,P,S1) => Adds(S0,P) & Precedes(S0,S1)
Link(S0,R,S1) => Adds(S0,R) & Precedes(S0,S1)
Link(S0,P,S2) => Adds(S0,P) & Precedes(S0,S2)
Link(S0,R,S2) => Adds(S0,R) & Precedes(S0,S2)
Link(S1,P,S2) => Adds(S1,P) & Precedes(S1,S2)
Link(S1,R,S2) => Adds(S1,R) & Precedes(S1,S2)
Link(S1,Q,S8) => Adds(S1,Q) & Precedes(S1,S8)
Link(S2,Q,S8) => Adds(S2,Q) & Precedes(S2,S8)
Link(S1,W,S8) => Adds(S1,W) & Precedes(S1,S8)
Link(S2,W,S8) => Adds(S2,W) & Precedes(S2,S8)

Link implies preservation by intervening steps:

Link(S1,Q,S8) & Deletes(S2,Q) => Precedes (S2,S1) V Precedes(S8,S2)
Link(S2,Q,S8) & Deletes(S1,Q) => Precedes (S1,S2) V Precedes(S8,S1)