Multi Agent Systems - Question 3

3.

a) We used the following query to verify whether free6pm(b) belonged to the grounded set.

grounded((free6pm_b, _)).

The query returned *yes* and therefore confirming that free6pm(b) belonged to the grounded set.

Alternatively, we also ran the query, *grounded(X)*, where X is bound to the result in the form of (claim, [assumptions]). This returned the grounded set as follows.

X = (free6pm_a,[free6pm_a]) ? ;
X = (free6pm_b,[free6pm_b]) ? ;
X = (free8am_b,[free8am_b]) ? ;
X = (child_a,[child_a]) ? ;
X = (overweight_b,[overweight_b]) ? ;
X = (not_get8am_a,[not_get8am_a]) ? ;
X = (not_free8am_a,[child_a]) ? ;
X = (not_sports_b,[overweight_b]) ? ;

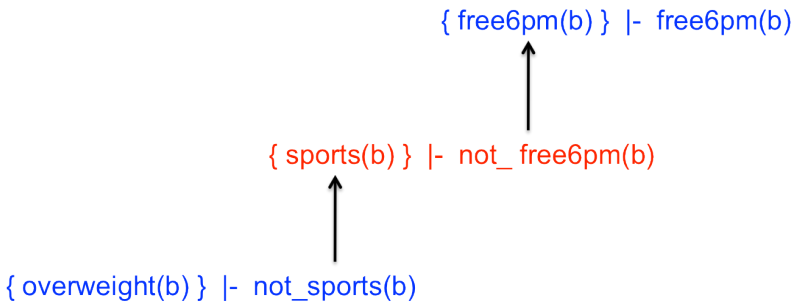
This once again confirmed that the claim free6pm(b) belonged to the grounded set as the argument *(free6pm_b,[free6pm_b])* is grounded.

To run this query, we had to modify our implementation of grounded to take into account complex loops as this example dealt with a larger cycle.

b) SXDD confirms that free6pm(b) belongs to the grounded extension. The table below shows the AB dispute derivation table for free6pm(b).

P: Proponent	O: Opponent	D: Assumptions Supporting Proponent	C: Culprits Chosen in Opponent	Explanation
{free6pm(b)}	{ }	{free6pm(b)}	{ }	Proponent's initial argument which itself is an assumption
{ }	{{not_free6pm(b)}}	{free6pm(b)}	{ }	Claim which is contrary of Proponent's argument
{ }	{{sports(b)}}	{free6pm(b)}	{ }	Derived assumption supporting the Opponent's claim
{not_sports(b)}	{{ }}	{free6pm(b)}	{sports(b)}	Claim which is contrary of Opponent's assumption
{overweight(b)}	{{ }}	{free6pm(b), overweight(b)}	{sports(b)}	Derived assumption supporting the Proponent's claim
{ }	{not_overweight(b)}	{free6pm(b), overweight(b)}	{sports(b)}	Claim which is contrary of Proponent's argument but is not supported by any assumptions, so Opponent cannot validly attack the Proponent with the claim, leaving the Proponent the winner.

This illustrates the dispute derivation.



c) According to the grounded extension, it was not able to deduce which appointments Anne and Boris would get. Although it was correctly able to infer that Boris would not get the 8am.

With the stable semantics, there could be two possible stable sets.

1. Boris gets the 8am and Anne gets the 6pm.
{Grounded} U {get6pm(a), get8am(b), not_get6pm(b)}

We included them into the stable set because:

- get6pm(a) attacks not_get6pm(a) and get6pm(b)
- get8am(b) attacks not_get8am(b), get8am(a) and get6pm(b)
- not_get6pm(b) is attacked by get6pm(b) but if we include get6pm(b) in the stable set it would attack get6pm(a) inside the stable set and thus break the stable semantics. Therefore we include not_get6pm(b)

2. Boris gets the 6pm and Anne doesn't get an appointment
{Grounded} U {get6pm(b), not_get6pm(a), not_get8am(b)}

We included them into the stable set because:

- get6pm(b) attacks not_get6pm(b), get6pm(a), get8am(b)
- not_get6pm(a) is attacked by get6pm(a) but if we include get6pm(a) in the stable set it would attack get6pm(b) inside the stable set and thus break the stable semantics. Therefore we include not_get6pm(a)
- not_get8am(b) is attacked by get8am(b) but if we include get8am(b) in the stable set it would attack get6pm(b) inside the stable set and thus break the stable semantics. Therefore we include not_get8am(b)

We can see that with stable semantics we (an agent) can strongly infer who gets what appointments (in the two different cases). However if we only look at the grounded extension, we can only see what must hold in all cases, which in our case is not_get8am(a), i.e. Anne does not get the 8am appointment. From this we cannot infer who gets what appointment, for example both of them may not get any appointment which would hold according to our grounded extension.