

1. The idea of common knowledge is important because it is able to extend the idea of collective knowledge (where everybody knows ϕ) with recursion, so that everybody knows that ... everybody knows ϕ . This creates a new way of reasoning within a group, as more information is available to each individual about what is known. In particular it helps with simultaneous decision making.

In the paper it is stated that this idea of common knowledge makes driving on the road possible (because everybody has to know that everybody else knows the rule to even dare to drive). Another possible application is in case of modeling negotiation. If I know that you know how much an item is worth, and you know that, it seems both parties will simultaneously get to a fair price.

2. For proposition p to be common knowledge it has to be so that given a state x , $(M, x) \models C_G p$ where $G = Agent1, Agent2$. So both need to know that p is the case in a state x and both have to know that both know that this is the case etc. (as can be derived from the definitions of page 43)

In state t , p is not common knowledge as *agent1* considers both t and s possible (he cannot see the difference between both). Although he knows that *agent2* knows, this is not sufficient.

In state s , p is no common knowledge either, as shown with the reasoning on page 42 (again *agent1* cannot differentiate between s and t , which both have a different value for p).

In state u , *agent2* considers both s and t possible, but since in both worlds p is true, he knows p . *Agent1* only considers u possible so he knows p is true as well. They also both know that about each other, so in this state there is common knowledge about P .

Unfortunately, I do not understand the tautology question, so I will not be able to answer that.

- 3.

$$(M, s) \models C_G \varphi \Leftrightarrow (M, s) \models E_G(\varphi \wedge C_G \phi) \quad (1)$$

Can be rewritten as:

$$(M, s) \models C_G \varphi \Leftrightarrow (M, s) \models E_G(\varphi) \wedge E_G(C_G \varphi) \quad (2)$$

Because of the definition on page 43 this can be rewritten as:

$$(M, s) \models C_G \varphi \Leftrightarrow (M, s) \models E_G(\varphi) \wedge E_G(E_G^{k-1} \varphi) \text{ for } k = 2, 3, \dots \quad (3)$$

When we set $(M, s) \models C_G \varphi$ to true this can be rewritten as:

$$(M, s) \models E_G^k \varphi \text{ for } k = 1, 2, \dots \Leftrightarrow (M, s) \models E_G(\varphi) \wedge E_G(E_G^{k-1} \varphi) \text{ for } k = 2, 3, \dots \quad (4)$$

More rewriting:

$$(M, s) \models E_G^k \varphi \text{ for } k = (1, 2, \dots) \Leftrightarrow (M, s) \models E_G(\varphi) \wedge E_G^k \varphi \text{ for } k = 1, 2, \dots \quad (5)$$

If E_G^k is true then $E_G(\varphi)$ as well, so that:

$$(M, s) \models E_G^k \varphi \text{ for } k = 1, 2, \dots \Leftrightarrow (M, s) \models E_G^k \varphi \text{ for } k = 1, 2, \dots \quad (6)$$

In case $(M, s) \models C_G \varphi$ is not set to true the derivation could not be made, hence the case would be that $(M, s) \not\models E_G(\varphi \wedge C_G \varphi)$

4. For an attack to happen not only does the second commander have to receive the messenger, but also does the first commander need to know the second commander knows that he wants to plan the attack and agrees to it. The second commander then needs to know that the first commander knows this and so on. Common knowledge in this case would enable the attack happening as then both parties would know that the other party knew indefinitely.

5. *Synchronous* in this problem means that all nonfaulty(not lying) generals after the same amount of time choose for the same time to attack

Searching for agreement is the process for the nonfaulty processors to agree upon one and the same strategy, while the fault processors in *Byzantine failure* mode try to confuse these faulty processors without their knowledge.

Point to point means that all generals are fully connected to all other generals, and so is the messaging.

6. In the paper is stated that always SBA can be made, but in the case of *Byzantine failure* mode for faulty processors there have to be at least $n = 3 * f + 1$ more nonfaulty processors than faulty ones (where n is nonfaulty and f is faulty), else no agreement can be made. This is only true when signatures can be forgeable (if a faulty processor can seem to be an unfaulty one), else SBA can be made with arbitrary many faulty processors. The SBA can be made in $f+1$ rounds, where f is the upper bound of faulty processors in the system.
7. From theorem 6.1 we can deduce that $(R, r, t) \models \varphi$ means that at a possible time t of run r φ follows. Now $(R, r, t) \models C_N \varphi$ means that at time t from run r it is common knowledge for all members of group N that φ is true. This is if and only if every member of N knows that φ and knows that φ is common knowledge (as is shown in exercise 3).
8. So in time t of run r φ is common knowledge of group N , or in other words, all members of N know that all members of N know that all members of N know... etc. that φ . A possible example, that might not be the most interesting I admit is when a group of N people pick a card from the deck and shows it to the rest of the group. The card distribution is now common knowledge at a given interval in this specific run (the next time interval the cards might be turned down and shuffled). The group of N people that show their card can be extended with "faulty people" who keep their card hidden and thus shows to the rest which are faulty and which are not.
9.
 - A *knowledge-based protocol* is a protocol wherein the processors explicitly act on the knowledge they have. The next action is solely based on that.
 - A *Crash failure* occurs when a faulty processor no longer transmits its messages, and thus does no longer communicate with the rest of the processors. In the traditional Byzantine agreement task this could be messengers that were intercepted and killed.

An *Omission failure* occurs when a faulty processor only some of the time does not deliver its messages to other processors. In the traditional task this could be lazy messengers who do not take their jobs seriously.

- *Byzantine Errors* occurs when faulty processors are not following protocol (or crash failure or omission failure occurs), like the lying generals in the traditional task. This is extra hard because there is not an absence of information, but wrong information spread around in the system, and when forgery is possible it takes longer for the nonfaulty systems to find agreement about their value.

10. I'll use p_i for processors on place $i \in 1, 2, 3, 4$ In case of crash failure:

For 0000: First round:

p_1 sends data to all other processors

p_2 sends data to all other processors

p_3 fails and does not send data

p_4 sends data to all other processors

After first round system halts and all nonfaulty processors decide on 0

For 1111: First round:

p_1 sends data to all other processors

p_2 sends data to all other processors

p_3 fails and does not send data

p_4 sends data to all other processors

After first round system halts and all nonfaulty processors decide on 1

11. I assume that a *standard* protocol correspond to the basic definition as we learned in the multi agent slides, where a protocol determines an action for each state.

Halpern states that *If we restrict our attention to crash failures or omission failures, there are only finitely many possible global states at any time, so that we can calculate the truth of a fact at any point simply by applying the definition of \models given in Section 2.* I find this a rather unhelpful explanation as I have no idea what a \models definition is.