|  |  |
| --- | --- |
| **Jimmy Holdö**  **890130-6319**  **gusholji@student.gu.se** | **Henrik Möller**  **xxxxxx-xxxx**  **…@...** |

**Question: Can you define the limited version in terms of the unlimited one?**

**Yes, this is possible because Haskell is a lazy language. Because of this the next instruction is not available before it is needed and if we set a limit after the x:et instruction we have created a finite version. Where the infinite version has only been instantiated to the point where it is needed.**

**Question: What definition of time do you use (what can a turtle achieve in a single time unit)?**

**The first thought was to only give time cost for instruction that resulted in visual changes on the screen but we realized soon that it was not ideal. For example if we have limited 20 (forever(color rgb) then the program would run forever. We decided therefore that everything should cost one time step and instruction that result in visual changes on the screen cost two time steps.**

**Questions: What happens after a parallel composition finishes? Is your parallel composition commutative, is it associative? (To answer this question you must first define what it means for programs to be equal.) What happens if a turtle runs forever only turning left in parallel with another turtle running the spiral example? Does your textual interface handle this situation correctly, if not - how would you fix it?**

**Our language is not defined for a sequence of a parallel composition and the something else because there does not exist a logical point from where the next instruction shall start from. If we should for example have started the next instruction from the right branch, then it is equal to a sequence of the right branch only. There is also the problem if a branch is infinite. Because of this we decided that when a parallel composition is finished then the program is finished.**

**So when a parallel composition p<|>q is launched, p and q does not have any dependences on each other. Therefore, the result is going to be the same if we launch q<|>p.**

**Question: How does parallel composition interact with lifespan and limited? (lifespan does not need to correspond realistically to actual life spans, just specify how it works.)**

**The thought was that if a lifespan or a limit is given before the program splits then both programs is given the life or limit the turtle had before the split. If one branch is given a lifespan or limit inside the branch, then it does not affect the other branch. This is the goal but somewhere we have a bug that we have not found that does so that one branch affect the other. It does not happen always but in some cases it occurs.**