

15-317 Homework 11

April 21, 2022

Task 3 (10 pts). Verify that tensor \otimes satisfies local soundness and completeness.

Solution 3:



Task 4 (10 pts). Verify that that $\mathbf{1}$ satisfies local soundness and completeness.

Solution 4:



Task 5 (10 pts). Verify that \top satisfies local soundness and completeness.

Solution 5:



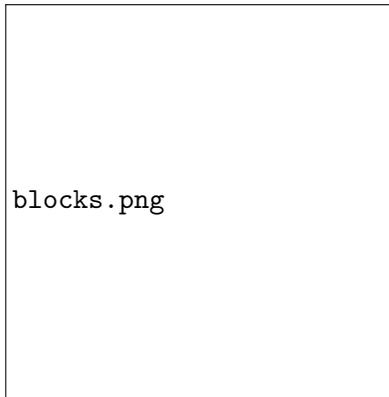
Task 6 (10 pts). So far we have assumed that the table is infinitely broad and can therefore accommodate any number of blocks. **Now consider the case that the table in fact only has a finite number of spaces for blocks on it, and modify the axioms of Blocks World above in order to preserve this invariant.**

You should use an atomic predicate `space`, which represents one open horizontal space on the table. A correct solution to this task will not depend on the size of the table.

Solution 6:

□

Task 7 (10 pts). Consider the following Blocks World scenario:



Write a formula in linear logic which expresses this configuration, assuming that **four** blocks can fit directly on the table.

Solution 7:

□

Task 8 (10 pts). Assuming *cut*, show that these two versions of the left rule are equivalent. That is, show that $\backslash L'$ is a derived rule for the sequent calculus which has just $\backslash L$, and show that $\backslash L$ is a derived rule for the sequent calculus which has just $\backslash L'$.

Solution 8:

□

Task 9 (10 pts). As is the case with many seemingly sensible rules, the new left rule $\backslash L'$ will actually destroy the logical character of the sequent calculus. In particular, the version of sequent calculus with $\backslash L'$ and without $\backslash L$ does not enjoy the admissibility of *cut*, and thence does not validate the cut elimination theorem.

Demonstrate a counterexample to cut elimination in the calculus with $\backslash L'$; to be precise, this should be a sequent $\Gamma \Longrightarrow C$ for some specific Γ and specific C which can be proved with *cut* but has no cut-free proof.

Solution 9:

□

Task 10 (10 pts). Use *ordered inference rules* to encode a procedure that decides whether a binary string contains an even or odd number of 1-bits in it.

Specifically, introducing a new atomic proposition `par` together with rules such that when \vec{A} is the encoding of a binary string in standard form, one can derive $\frac{\vec{A}}{\$} \text{par}$ iff the string \vec{A} has an even number of 1-bits, and one can derive $\frac{\vec{A}}{\$} \text{b1}$ iff \vec{A} has an odd number of 1-bits.

You may freely introduce any auxiliary propositions and rules that you require.

Solution 10:

□

Task 11 (10 pts). Rewrite your solution to the previous task in the form of a well-typed ordered concurrent program in the subsingleton fragment. You may freely make any definitions you require, including recursive definitions of session types and concurrent programs.

Your solution should include the definition of a type of bit strings *bits*, together with a program `par` that has the type $\text{bits} \vdash \text{par} : \text{bits}$.

Solution 11:

□