This file contains the appendices to a paper “An Automated Deduction of the Halting Problem”.

**APPENDIX 1. VAMPIRE proof of “The halting problem cannot be solved on a Turing machine”.** **See Sutcliffe 2008; Sutcliffe, Zimmer, and Schulz 2003, 2004; Sutcliffe and Suttner 2021 for syntax and ontology details.**

**Command line:**

$ ../bin/vampire\_rel.exe COM003+1.p > COM003+1.p.txt

**Proof.** The proof is by refutation. Such a proof first asserts the axioms/assumptions of the theorem of interest, and asserts the negation of the consequent that theorem, then proceeds to draw a contradiction.

**% Refutation found. Thanks to Tanya!**

**% SZS status Theorem for COM003+1**

**% SZS output start Proof for COM003+1**

**1. ? [X0] : (! [X1] : (program(X1) => ! [X2] : decides(X0,X1,X2)) & algorithm(X0)) => ? [X3] : (! [X1] : (program(X1) => ! [X2] : decides(X3,X1,X2)) & program(X3)) [input]**

**2. ! [X3] : ((! [X1] : (program(X1) => ! [X2] : decides(X3,X1,X2)) & program(X3)) => ! [X1,X2] : (((~halts2(X1,X2) & program(X1)) => (outputs(X3,bad) & halts3(X3,X1,X2))) & ((halts2(X1,X2) & program(X1)) => (outputs(X3,good) & halts3(X3,X1,X2))))) [input]**

**3. ? [X3] : (! [X1] : (((~halts2(X1,X1) & program(X1)) => (outputs(X3,bad) & halts3(X3,X1,X1))) & ((halts2(X1,X1) & program(X1)) => (outputs(X3,good) & halts3(X3,X1,X1)))) & program(X3)) => ? [X4] : (! [X1] : (((~halts2(X1,X1) & program(X1)) => (outputs(X4,bad) & halts2(X4,X1))) & ((halts2(X1,X1) & program(X1)) => (outputs(X4,good) & halts2(X4,X1)))) & program(X4)) [input]**

**4. ? [X4] : (! [X1] : (((~halts2(X1,X1) & program(X1)) => (outputs(X4,bad) & halts2(X4,X1))) & ((halts2(X1,X1) & program(X1)) => (outputs(X4,good) & halts2(X4,X1)))) & program(X4)) => ? [X5] : (! [X1] : (((~halts2(X1,X1) & program(X1)) => (outputs(X5,bad) & halts2(X5,X1))) & ((halts2(X1,X1) & program(X1)) => ~halts2(X5,X1))) & program(X5)) [input]**

**5. ~? [X6] : (! [X7] : (program(X7) => ! [X8] : decides(X6,X7,X8)) & algorithm(X6)) [input]**

**6. ~~? [X6] : (! [X7] : (program(X7) => ! [X8] : decides(X6,X7,X8)) & algorithm(X6)) [negated conjecture 5]**

**7. ? [X0] : (! [X1] : (program(X1) => ! [X2] : decides(X0,X1,X2)) & algorithm(X0)) => ? [X3] : (! [X4] : (program(X4) => ! [X5] : decides(X3,X4,X5)) & program(X3)) [rectify 1]**

**8. ! [X0] : ((! [X1] : (program(X1) => ! [X2] : decides(X0,X1,X2)) & program(X0)) => ! [X3,X4] : (((~halts2(X3,X4) & program(X3)) => (outputs(X0,bad) & halts3(X0,X3,X4))) & ((halts2(X3,X4) & program(X3)) => (outputs(X0,good) & halts3(X0,X3,X4))))) [rectify 2]**

**9. ? [X0] : (! [X1] : (((~halts2(X1,X1) & program(X1)) => (outputs(X0,bad) & halts3(X0,X1,X1))) & ((halts2(X1,X1) & program(X1)) => (outputs(X0,good) & halts3(X0,X1,X1)))) & program(X0)) => ? [X2] : (! [X3] : (((~halts2(X3,X3) & program(X3)) => (outputs(X2,bad) & halts2(X2,X3))) & ((halts2(X3,X3) & program(X3)) => (outputs(X2,good) & halts2(X2,X3)))) & program(X2)) [rectify 3]**

**10. ? [X0] : (! [X1] : (((~halts2(X1,X1) & program(X1)) => (outputs(X0,bad) & halts2(X0,X1))) & ((halts2(X1,X1) & program(X1)) => (outputs(X0,good) & halts2(X0,X1)))) & program(X0)) => ? [X2] : (! [X3] : (((~halts2(X3,X3) & program(X3)) => (outputs(X2,bad) & halts2(X2,X3))) & ((halts2(X3,X3) & program(X3)) => ~halts2(X2,X3))) & program(X2)) [rectify 4]**

**11. ~~? [X0] : (! [X1] : (program(X1) => ! [X2] : decides(X0,X1,X2)) & algorithm(X0)) [rectify 6]**

**12. ? [X0] : (! [X1] : (program(X1) => ! [X2] : decides(X0,X1,X2)) & algorithm(X0)) [flattening 11]**

**13. ? [X3] : (! [X4] : (! [X5] : decides(X3,X4,X5) | ~program(X4)) & program(X3)) | ! [X0] : (? [X1] : (? [X2] : ~decides(X0,X1,X2) & program(X1)) | ~algorithm(X0)) [ennf transformation 7]**

**14. ! [X0] : (! [X3,X4] : (((outputs(X0,bad) & halts3(X0,X3,X4)) | (halts2(X3,X4) | ~program(X3))) & ((outputs(X0,good) & halts3(X0,X3,X4)) | (~halts2(X3,X4) | ~program(X3)))) | (? [X1] : (? [X2] : ~decides(X0,X1,X2) & program(X1)) | ~program(X0))) [ennf transformation 8]**

**15. ! [X0] : (! [X3,X4] : (((outputs(X0,bad) & halts3(X0,X3,X4)) | halts2(X3,X4) | ~program(X3)) & ((outputs(X0,good) & halts3(X0,X3,X4)) | ~halts2(X3,X4) | ~program(X3))) | ? [X1] : (? [X2] : ~decides(X0,X1,X2) & program(X1)) | ~program(X0)) [flattening 14]**

**16. ? [X2] : (! [X3] : (((outputs(X2,bad) & halts2(X2,X3)) | (halts2(X3,X3) | ~program(X3))) & ((outputs(X2,good) & halts2(X2,X3)) | (~halts2(X3,X3) | ~program(X3)))) & program(X2)) | ! [X0] : (? [X1] : (((~outputs(X0,bad) | ~halts3(X0,X1,X1)) & (~halts2(X1,X1) & program(X1))) | ((~outputs(X0,good) | ~halts3(X0,X1,X1)) & (halts2(X1,X1) & program(X1)))) | ~program(X0)) [ennf transformation 9]**

**17. ? [X2] : (! [X3] : (((outputs(X2,bad) & halts2(X2,X3)) | halts2(X3,X3) | ~program(X3)) & ((outputs(X2,good) & halts2(X2,X3)) | ~halts2(X3,X3) | ~program(X3))) & program(X2)) | ! [X0] : (? [X1] : (((~outputs(X0,bad) | ~halts3(X0,X1,X1)) & ~halts2(X1,X1) & program(X1)) | ((~outputs(X0,good) | ~halts3(X0,X1,X1)) & halts2(X1,X1) & program(X1))) | ~program(X0)) [flattening 16]**

**18. ? [X2] : (! [X3] : (((outputs(X2,bad) & halts2(X2,X3)) | (halts2(X3,X3) | ~program(X3))) & (~halts2(X2,X3) | (~halts2(X3,X3) | ~program(X3)))) & program(X2)) | ! [X0] : (? [X1] : (((~outputs(X0,bad) | ~halts2(X0,X1)) & (~halts2(X1,X1) & program(X1))) | ((~outputs(X0,good) | ~halts2(X0,X1)) & (halts2(X1,X1) & program(X1)))) | ~program(X0)) [ennf transformation 10]**

**19. ? [X2] : (! [X3] : (((outputs(X2,bad) & halts2(X2,X3)) | halts2(X3,X3) | ~program(X3)) & (~halts2(X2,X3) | ~halts2(X3,X3) | ~program(X3))) & program(X2)) | ! [X0] : (? [X1] : (((~outputs(X0,bad) | ~halts2(X0,X1)) & ~halts2(X1,X1) & program(X1)) | ((~outputs(X0,good) | ~halts2(X0,X1)) & halts2(X1,X1) & program(X1))) | ~program(X0)) [flattening 18]**

**20. ? [X0] : (! [X1] : (! [X2] : decides(X0,X1,X2) | ~program(X1)) & algorithm(X0)) [ennf transformation 12]**

**21. ! [X0,X1] : (((~outputs(X0,good) | ~halts3(X0,X1,X1)) & halts2(X1,X1) & program(X1)) | ~sP0(X0,X1)) [predicate definition introduction]**

**22. ? [X2] : (! [X3] : (((outputs(X2,bad) & halts2(X2,X3)) | halts2(X3,X3) | ~program(X3)) & ((outputs(X2,good) & halts2(X2,X3)) | ~halts2(X3,X3) | ~program(X3))) & program(X2)) | ~sP1 [predicate definition introduction]**

**23. sP1 | ! [X0] : (? [X1] : (((~outputs(X0,bad) | ~halts3(X0,X1,X1)) & ~halts2(X1,X1) & program(X1)) | sP0(X0,X1)) | ~program(X0)) [definition folding 17,22,21]**

**24. ! [X0,X1] : (((~outputs(X0,good) | ~halts2(X0,X1)) & halts2(X1,X1) & program(X1)) | ~sP2(X0,X1)) [predicate definition introduction]**

**25. ? [X2] : (! [X3] : (((outputs(X2,bad) & halts2(X2,X3)) | halts2(X3,X3) | ~program(X3)) & (~halts2(X2,X3) | ~halts2(X3,X3) | ~program(X3))) & program(X2)) | ~sP3 [predicate definition introduction]**

**26. sP3 | ! [X0] : (? [X1] : (((~outputs(X0,bad) | ~halts2(X0,X1)) & ~halts2(X1,X1) & program(X1)) | sP2(X0,X1)) | ~program(X0)) [definition folding 19,25,24]**

**27. ? [X0] : (! [X1] : (! [X2] : decides(X0,X1,X2) | ~program(X1)) & program(X0)) | ! [X3] : (? [X4] : (? [X5] : ~decides(X3,X4,X5) & program(X4)) | ~algorithm(X3)) [rectify 13]**

**28. ? [X0] : (! [X1] : (! [X2] : decides(X0,X1,X2) | ~program(X1)) & program(X0)) => (! [X1] : (! [X2] : decides(sK4,X1,X2) | ~program(X1)) & program(sK4)) [choice axiom]**

**29. ! [X3] : (? [X4] : (? [X5] : ~decides(X3,X4,X5) & program(X4)) => (? [X5] : ~decides(X3,sK5(X3),X5) & program(sK5(X3)))) [choice axiom]**

**30. ! [X3] : (? [X5] : ~decides(X3,sK5(X3),X5) => ~decides(X3,sK5(X3),sK6(X3))) [choice axiom]**

**31. (! [X1] : (! [X2] : decides(sK4,X1,X2) | ~program(X1)) & program(sK4)) | ! [X3] : ((~decides(X3,sK5(X3),sK6(X3)) & program(sK5(X3))) | ~algorithm(X3)) [skolemisation 27,30,29,28]**

**32. ! [X0] : (! [X1,X2] : (((outputs(X0,bad) & halts3(X0,X1,X2)) | halts2(X1,X2) | ~program(X1)) & ((outputs(X0,good) & halts3(X0,X1,X2)) | ~halts2(X1,X2) | ~program(X1))) | ? [X3] : (? [X4] : ~decides(X0,X3,X4) & program(X3)) | ~program(X0)) [rectify 15]**

**33. ! [X0] : (? [X3] : (? [X4] : ~decides(X0,X3,X4) & program(X3)) => (? [X4] : ~decides(X0,sK7(X0),X4) & program(sK7(X0)))) [choice axiom]**

**34. ! [X0] : (? [X4] : ~decides(X0,sK7(X0),X4) => ~decides(X0,sK7(X0),sK8(X0))) [choice axiom]**

**35. ! [X0] : (! [X1,X2] : (((outputs(X0,bad) & halts3(X0,X1,X2)) | halts2(X1,X2) | ~program(X1)) & ((outputs(X0,good) & halts3(X0,X1,X2)) | ~halts2(X1,X2) | ~program(X1))) | (~decides(X0,sK7(X0),sK8(X0)) & program(sK7(X0))) | ~program(X0)) [skolemisation 32,34,33]**

**36. ? [X2] : (! [X3] : (((outputs(X2,bad) & halts2(X2,X3)) | halts2(X3,X3) | ~program(X3)) & ((outputs(X2,good) & halts2(X2,X3)) | ~halts2(X3,X3) | ~program(X3))) & program(X2)) | ~sP1 [nnf transformation 22]**

**37. ? [X0] : (! [X1] : (((outputs(X0,bad) & halts2(X0,X1)) | halts2(X1,X1) | ~program(X1)) & ((outputs(X0,good) & halts2(X0,X1)) | ~halts2(X1,X1) | ~program(X1))) & program(X0)) | ~sP1 [rectify 36]**

**38. ? [X0] : (! [X1] : (((outputs(X0,bad) & halts2(X0,X1)) | halts2(X1,X1) | ~program(X1)) & ((outputs(X0,good) & halts2(X0,X1)) | ~halts2(X1,X1) | ~program(X1))) & program(X0)) => (! [X1] : (((outputs(sK9,bad) & halts2(sK9,X1)) | halts2(X1,X1) | ~program(X1)) & ((outputs(sK9,good) & halts2(sK9,X1)) | ~halts2(X1,X1) | ~program(X1))) & program(sK9)) [choice axiom]**

**39. (! [X1] : (((outputs(sK9,bad) & halts2(sK9,X1)) | halts2(X1,X1) | ~program(X1)) & ((outputs(sK9,good) & halts2(sK9,X1)) | ~halts2(X1,X1) | ~program(X1))) & program(sK9)) | ~sP1 [skolemisation 37,38]**

**40. ! [X0,X1] : (((~outputs(X0,good) | ~halts3(X0,X1,X1)) & halts2(X1,X1) & program(X1)) | ~sP0(X0,X1)) [nnf transformation 21]**

**41. ! [X0] : (? [X1] : (((~outputs(X0,bad) | ~halts3(X0,X1,X1)) & ~halts2(X1,X1) & program(X1)) | sP0(X0,X1)) => (((~outputs(X0,bad) | ~halts3(X0,sK10(X0),sK10(X0))) & ~halts2(sK10(X0),sK10(X0)) & program(sK10(X0))) | sP0(X0,sK10(X0)))) [choice axiom]**

**42. sP1 | ! [X0] : ((((~outputs(X0,bad) | ~halts3(X0,sK10(X0),sK10(X0))) & ~halts2(sK10(X0),sK10(X0)) & program(sK10(X0))) | sP0(X0,sK10(X0))) | ~program(X0)) [skolemisation 23,41]**

**43. ? [X2] : (! [X3] : (((outputs(X2,bad) & halts2(X2,X3)) | halts2(X3,X3) | ~program(X3)) & (~halts2(X2,X3) | ~halts2(X3,X3) | ~program(X3))) & program(X2)) | ~sP3 [nnf transformation 25]**

**44. ? [X0] : (! [X1] : (((outputs(X0,bad) & halts2(X0,X1)) | halts2(X1,X1) | ~program(X1)) & (~halts2(X0,X1) | ~halts2(X1,X1) | ~program(X1))) & program(X0)) | ~sP3 [rectify 43]**

**45. ? [X0] : (! [X1] : (((outputs(X0,bad) & halts2(X0,X1)) | halts2(X1,X1) | ~program(X1)) & (~halts2(X0,X1) | ~halts2(X1,X1) | ~program(X1))) & program(X0)) => (! [X1] : (((outputs(sK11,bad) & halts2(sK11,X1)) | halts2(X1,X1) | ~program(X1)) & (~halts2(sK11,X1) | ~halts2(X1,X1) | ~program(X1))) & program(sK11)) [choice axiom]**

**46. (! [X1] : (((outputs(sK11,bad) & halts2(sK11,X1)) | halts2(X1,X1) | ~program(X1)) & (~halts2(sK11,X1) | ~halts2(X1,X1) | ~program(X1))) & program(sK11)) | ~sP3 [skolemisation 44,45]**

**47. ! [X0,X1] : (((~outputs(X0,good) | ~halts2(X0,X1)) & halts2(X1,X1) & program(X1)) | ~sP2(X0,X1)) [nnf transformation 24]**

**48. ! [X0] : (? [X1] : (((~outputs(X0,bad) | ~halts2(X0,X1)) & ~halts2(X1,X1) & program(X1)) | sP2(X0,X1)) => (((~outputs(X0,bad) | ~halts2(X0,sK12(X0))) & ~halts2(sK12(X0),sK12(X0)) & program(sK12(X0))) | sP2(X0,sK12(X0)))) [choice axiom]**

**49. sP3 | ! [X0] : ((((~outputs(X0,bad) | ~halts2(X0,sK12(X0))) & ~halts2(sK12(X0),sK12(X0)) & program(sK12(X0))) | sP2(X0,sK12(X0))) | ~program(X0)) [skolemisation 26,48]**

**50. ? [X0] : (! [X1] : (! [X2] : decides(X0,X1,X2) | ~program(X1)) & algorithm(X0)) => (! [X1] : (! [X2] : decides(sK13,X1,X2) | ~program(X1)) & algorithm(sK13)) [choice axiom]**

**51. ! [X1] : (! [X2] : decides(sK13,X1,X2) | ~program(X1)) & algorithm(sK13) [skolemisation 20,50]**

**52. program(sK4) | program(sK5(X3)) | ~algorithm(X3) [cnf transformation 31]**

**53. program(sK4) | ~decides(X3,sK5(X3),sK6(X3)) | ~algorithm(X3) [cnf transformation 31]**

**54. decides(sK4,X1,X2) | ~program(X1) | program(sK5(X3)) | ~algorithm(X3) [cnf transformation 31]**

**55. decides(sK4,X1,X2) | ~program(X1) | ~decides(X3,sK5(X3),sK6(X3)) | ~algorithm(X3) [cnf transformation 31]**

**56. halts3(X0,X1,X2) | ~halts2(X1,X2) | ~program(X1) | program(sK7(X0)) | ~program(X0) [cnf transformation 35]**

**57. halts3(X0,X1,X2) | ~halts2(X1,X2) | ~program(X1) | ~decides(X0,sK7(X0),sK8(X0)) | ~program(X0) [cnf transformation 35]**

**58. outputs(X0,good) | ~halts2(X1,X2) | ~program(X1) | program(sK7(X0)) | ~program(X0) [cnf transformation 35]**

**59. outputs(X0,good) | ~halts2(X1,X2) | ~program(X1) | ~decides(X0,sK7(X0),sK8(X0)) | ~program(X0) [cnf transformation 35]**

**60. halts3(X0,X1,X2) | halts2(X1,X2) | ~program(X1) | program(sK7(X0)) | ~program(X0) [cnf transformation 35]**

**61. ~decides(X0,sK7(X0),sK8(X0)) | halts2(X1,X2) | ~program(X1) | halts3(X0,X1,X2) | ~program(X0) [cnf transformation 35]**

**62. outputs(X0,bad) | halts2(X1,X2) | ~program(X1) | program(sK7(X0)) | ~program(X0) [cnf transformation 35]**

**63. outputs(X0,bad) | halts2(X1,X2) | ~program(X1) | ~decides(X0,sK7(X0),sK8(X0)) | ~program(X0) [cnf transformation 35]**

**64. program(sK9) | ~sP1 [cnf transformation 39]**

**65. halts2(sK9,X1) | ~halts2(X1,X1) | ~program(X1) | ~sP1 [cnf transformation 39]**

**66. outputs(sK9,good) | ~halts2(X1,X1) | ~program(X1) | ~sP1 [cnf transformation 39]**

**67. halts2(sK9,X1) | halts2(X1,X1) | ~program(X1) | ~sP1 [cnf transformation 39]**

**68. outputs(sK9,bad) | halts2(X1,X1) | ~program(X1) | ~sP1 [cnf transformation 39]**

**69. ~sP0(X0,X1) | program(X1) [cnf transformation 40]**

**70. ~sP0(X0,X1) | halts2(X1,X1) [cnf transformation 40]**

**71. ~halts3(X0,X1,X1) | ~outputs(X0,good) | ~sP0(X0,X1) [cnf transformation 40]**

**72. sP1 | program(sK10(X0)) | sP0(X0,sK10(X0)) | ~program(X0) [cnf transformation 42]**

**74. sP1 | ~outputs(X0,bad) | ~halts3(X0,sK10(X0),sK10(X0)) | sP0(X0,sK10(X0)) | ~program(X0) [cnf transformation 42]**

**75. program(sK11) | ~sP3 [cnf transformation 46]**

**76. ~halts2(sK11,X1) | ~halts2(X1,X1) | ~program(X1) | ~sP3 [cnf transformation 46]**

**77. halts2(sK11,X1) | halts2(X1,X1) | ~program(X1) | ~sP3 [cnf transformation 46]**

**79. ~sP2(X0,X1) | program(X1) [cnf transformation 47]**

**81. ~sP2(X0,X1) | ~halts2(X0,X1) | ~outputs(X0,good) [cnf transformation 47]**

**82. sP3 | program(sK12(X0)) | sP2(X0,sK12(X0)) | ~program(X0) [cnf transformation 49]**

**83. sP3 | ~halts2(sK12(X0),sK12(X0)) | sP2(X0,sK12(X0)) | ~program(X0) [cnf transformation 49]**

**84. sP3 | ~outputs(X0,bad) | ~halts2(X0,sK12(X0)) | sP2(X0,sK12(X0)) | ~program(X0) [cnf transformation 49]**

**85. algorithm(sK13) [cnf transformation 51]**

**86. decides(sK13,X1,X2) | ~program(X1) [cnf transformation 51]**

**88. 1 <=> ! [X3] : (~decides(X3,sK5(X3),sK6(X3)) | ~algorithm(X3)) [avatar definition]**

**89. ~decides(X3,sK5(X3),sK6(X3)) | ~algorithm(X3) <- (1) [avatar component clause 88]**

**91. 2 <=> ! [X1,X2] : (decides(sK4,X1,X2) | ~program(X1)) [avatar definition]**

**92. decides(sK4,X1,X2) | ~program(X1) <- (2) [avatar component clause 91]**

**93. 1 | 2 [avatar split clause 55,91,88]**

**95. 3 <=> ! [X3] : (program(sK5(X3)) | ~algorithm(X3)) [avatar definition]**

**96. program(sK5(X3)) | ~algorithm(X3) <- (3) [avatar component clause 95]**

**97. 3 | 2 [avatar split clause 54,91,95]**

**99. 4 <=> program(sK4) [avatar definition]**

**101. program(sK4) <- (4) [avatar component clause 99]**

**102. 1 | 4 [avatar split clause 53,99,88]**

**103. 3 | 4 [avatar split clause 52,99,95]**

**105. 5 <=> ! [X1,X2] : (halts2(X1,X2) | ~program(X1)) [avatar definition]**

**106. halts2(X1,X2) | ~program(X1) <- (5) [avatar component clause 105]**

**108. 6 <=> ! [X0] : (outputs(X0,bad) | ~program(X0) | ~decides(X0,sK7(X0),sK8(X0))) [avatar definition]**

**109. ~decides(X0,sK7(X0),sK8(X0)) | ~program(X0) | outputs(X0,bad) <- (6) [avatar component clause 108]**

**110. 5 | 6 [avatar split clause 63,108,105]**

**112. 7 <=> ! [X0] : (outputs(X0,bad) | ~program(X0) | program(sK7(X0))) [avatar definition]**

**113. outputs(X0,bad) | ~program(X0) | program(sK7(X0)) <- (7) [avatar component clause 112]**

**114. 5 | 7 [avatar split clause 62,112,105]**

**116. 8 <=> ! [X1,X2] : (~halts2(X1,X2) | ~program(X1)) [avatar definition]**

**117. ~halts2(X1,X2) | ~program(X1) <- (8) [avatar component clause 116]**

**119. 9 <=> ! [X0] : (outputs(X0,good) | ~program(X0) | ~decides(X0,sK7(X0),sK8(X0))) [avatar definition]**

**120. ~decides(X0,sK7(X0),sK8(X0)) | ~program(X0) | outputs(X0,good) <- (9) [avatar component clause 119]**

**121. 8 | 9 [avatar split clause 59,119,116]**

**123. 10 <=> ! [X0] : (outputs(X0,good) | ~program(X0) | program(sK7(X0))) [avatar definition]**

**124. outputs(X0,good) | ~program(X0) | program(sK7(X0)) <- (10) [avatar component clause 123]**

**125. 8 | 10 [avatar split clause 58,123,116]**

**126. ~decides(X0,sK7(X0),sK8(X0)) | ~program(X1) | halts3(X0,X1,X2) | ~program(X0) [subsumption resolution 57,61]**

**127. halts3(X0,X1,X2) | ~program(X1) | program(sK7(X0)) | ~program(X0) [subsumption resolution 56,60]**

**129. 11 <=> sP1 [avatar definition]**

**133. 12 <=> ! [X1] : (halts2(X1,X1) | ~program(X1)) [avatar definition]**

**134. halts2(X1,X1) | ~program(X1) <- (12) [avatar component clause 133]**

**136. 13 <=> outputs(sK9,bad) [avatar definition]**

**139. ~11 | 12 | 13 [avatar split clause 68,136,133,129]**

**141. 14 <=> ! [X1] : (halts2(sK9,X1) | ~program(X1) | halts2(X1,X1)) [avatar definition]**

**142. halts2(sK9,X1) | ~program(X1) | halts2(X1,X1) <- (14) [avatar component clause 141]**

**143. ~11 | 14 [avatar split clause 67,141,129]**

**145. 15 <=> ! [X1] : (~halts2(X1,X1) | ~program(X1)) [avatar definition]**

**146. ~halts2(X1,X1) | ~program(X1) <- (15) [avatar component clause 145]**

**148. 16 <=> outputs(sK9,good) [avatar definition]**

**150. outputs(sK9,good) <- (16) [avatar component clause 148]**

**151. ~11 | 15 | 16 [avatar split clause 66,148,145,129]**

**153. 17 <=> ! [X1] : (halts2(sK9,X1) | ~program(X1) | ~halts2(X1,X1)) [avatar definition]**

**154. ~halts2(X1,X1) | ~program(X1) | halts2(sK9,X1) <- (17) [avatar component clause 153]**

**155. ~11 | 17 [avatar split clause 65,153,129]**

**157. 18 <=> program(sK9) [avatar definition]**

**159. program(sK9) <- (18) [avatar component clause 157]**

**160. ~11 | 18 [avatar split clause 64,157,129]**

**162. 19 <=> ! [X0] : (~outputs(X0,bad) | ~program(X0) | sP0(X0,sK10(X0)) | ~halts3(X0,sK10(X0),sK10(X0))) [avatar definition]**

**163. ~halts3(X0,sK10(X0),sK10(X0)) | ~program(X0) | sP0(X0,sK10(X0)) | ~outputs(X0,bad) <- (19) [avatar component clause 162]**

**164. 19 | 11 [avatar split clause 74,129,162]**

**169. sP1 | program(sK10(X0)) | ~program(X0) [subsumption resolution 72,69]**

**171. 21 <=> ! [X0] : (program(sK10(X0)) | ~program(X0)) [avatar definition]**

**172. program(sK10(X0)) | ~program(X0) <- (21) [avatar component clause 171]**

**173. 21 | 11 [avatar split clause 169,129,171]**

**175. 22 <=> sP3 [avatar definition]**

**184. 24 <=> ! [X1] : (halts2(sK11,X1) | ~program(X1) | halts2(X1,X1)) [avatar definition]**

**185. halts2(sK11,X1) | halts2(X1,X1) | ~program(X1) <- (24) [avatar component clause 184]**

**186. ~22 | 24 [avatar split clause 77,184,175]**

**188. 25 <=> ! [X1] : (~halts2(sK11,X1) | ~program(X1) | ~halts2(X1,X1)) [avatar definition]**

**189. ~halts2(sK11,X1) | ~program(X1) | ~halts2(X1,X1) <- (25) [avatar component clause 188]**

**190. ~22 | 25 [avatar split clause 76,188,175]**

**192. 26 <=> program(sK11) [avatar definition]**

**194. program(sK11) <- (26) [avatar component clause 192]**

**195. ~22 | 26 [avatar split clause 75,192,175]**

**197. 27 <=> ! [X0] : (~outputs(X0,bad) | ~program(X0) | sP2(X0,sK12(X0)) | ~halts2(X0,sK12(X0))) [avatar definition]**

**198. ~halts2(X0,sK12(X0)) | ~program(X0) | sP2(X0,sK12(X0)) | ~outputs(X0,bad) <- (27) [avatar component clause 197]**

**199. 27 | 22 [avatar split clause 84,175,197]**

**201. 28 <=> ! [X0] : (~halts2(sK12(X0),sK12(X0)) | ~program(X0) | sP2(X0,sK12(X0))) [avatar definition]**

**202. ~halts2(sK12(X0),sK12(X0)) | ~program(X0) | sP2(X0,sK12(X0)) <- (28) [avatar component clause 201]**

**203. 28 | 22 [avatar split clause 83,175,201]**

**204. sP3 | program(sK12(X0)) | ~program(X0) [subsumption resolution 82,79]**

**206. 29 <=> ! [X0] : (program(sK12(X0)) | ~program(X0)) [avatar definition]**

**207. program(sK12(X0)) | ~program(X0) <- (29) [avatar component clause 206]**

**208. 29 | 22 [avatar split clause 204,175,206]**

**209. halts2(sK9,X1) | ~program(X1) <- (14, 17) [subsumption resolution 142,154]**

**213. ~program(X1) <- (5, 8) [subsumption resolution 117,106]**

**214. $false <- (4, 5, 8) [resolution 213,101]**

**216. $false <- (5, 8, 26) [resolution 213,194]**

**217. ~5 | ~8 | ~26 [avatar contradiction clause 216]**

**219. ~4 | ~5 | ~8 [avatar contradiction clause 214]**

**226. halts2(sK11,sK11) | ~program(sK11) <- (24) [factoring 185]**

**248. ~program(sK4) | outputs(sK4,good) | ~program(sK7(sK4)) <- (2, 9) [resolution 120,92]**

**261. 32 <=> program(sK7(sK4)) [avatar definition]**

**263. ~program(sK7(sK4)) <- (~32) [avatar component clause 261]**

**265. 33 <=> outputs(sK4,good) [avatar definition]**

**267. outputs(sK4,good) <- (33) [avatar component clause 265]**

**269. ~program(X0) | program(sK7(X1)) | ~program(X1) | ~outputs(X1,good) | ~sP0(X1,X0) [resolution 127,71]**

**270. program(sK7(X1)) | ~program(X1) | ~outputs(X1,good) | ~sP0(X1,X0) [subsumption resolution 269,69]**

**271. ~sP0(X1,X0) | ~program(X1) | program(sK7(X1)) <- (10) [subsumption resolution 270,124]**

**272. ~program(X0) | sP2(X0,sK12(X0)) | ~program(sK12(X0)) <- (5, 28) [resolution 202,106]**

**274. sP2(X0,sK12(X0)) | ~program(X0) <- (5, 28, 29) [subsumption resolution 272,207]**

**275. ~program(X0) | ~halts2(X0,sK12(X0)) | ~outputs(X0,good) <- (5, 28, 29) [resolution 274,81]**

**278. ~outputs(X0,good) | ~program(X0) <- (5, 28, 29) [subsumption resolution 275,106]**

**302. ~program(X2) | halts3(sK4,X2,X3) | ~program(sK4) | ~program(sK7(sK4)) <- (2) [resolution 126,92]**

**305. ~program(sK4) <- (5, 28, 29, 33) [resolution 267,278]**

**306. $false <- (4, 5, 28, 29, 33) [subsumption resolution 305,101]**

**307. ~4 | ~5 | ~28 | ~29 | ~33 [avatar contradiction clause 306]**

**310. 34 <=> ! [X3,X2] : (~program(X2) | halts3(sK4,X2,X3)) [avatar definition]**

**311. halts3(sK4,X2,X3) | ~program(X2) <- (34) [avatar component clause 310]**

**316. ~algorithm(sK13) | ~program(sK5(sK13)) <- (1) [resolution 89,86]**

**318. ~program(sK5(sK13)) <- (1) [subsumption resolution 316,85]**

**321. ~program(X1) | sP0(X1,sK10(X1)) | ~outputs(X1,bad) | ~program(sK10(X1)) | program(sK7(X1)) | ~program(X1) <- (19) [resolution 163,127]**

**322. ~program(X1) | sP0(X1,sK10(X1)) | ~outputs(X1,bad) | ~program(sK10(X1)) | program(sK7(X1)) <- (19) [duplicate literal removal 321]**

**324. ~algorithm(sK13) <- (1, 3) [resolution 318,96]**

**325. $false <- (1, 3) [subsumption resolution 324,85]**

**326. ~1 | ~3 [avatar contradiction clause 325]**

**336. ~program(X1) | ~outputs(X1,bad) | ~program(sK10(X1)) | program(sK7(X1)) <- (10, 19) [subsumption resolution 322,271]**

**337. ~program(X1) | ~outputs(X1,bad) | program(sK7(X1)) <- (10, 19, 21) [subsumption resolution 336,172]**

**346. 38 <=> halts2(sK11,sK11) [avatar definition]**

**348. halts2(sK11,sK11) <- (38) [avatar component clause 346]**

**349. ~26 | 38 | ~24 [avatar split clause 226,184,346,192]**

**350. ~program(sK4) | outputs(sK4,good) <- (2, 9, 10) [subsumption resolution 248,124]**

**351. 33 | ~4 | ~2 | ~9 | ~10 [avatar split clause 350,123,119,91,99,265]**

**352. ~program(X2) | halts3(sK4,X2,X3) | ~program(sK4) <- (2) [subsumption resolution 302,127]**

**353. ~4 | 34 | ~2 [avatar split clause 352,91,310,99]**

**354. program(sK7(X0)) | ~program(X0) <- (7, 10, 19, 21) [subsumption resolution 113,337]**

**355. ~program(sK4) <- (7, 10, 19, 21, ~32) [resolution 263,354]**

**356. $false <- (4, 7, 10, 19, 21, ~32) [subsumption resolution 355,101]**

**357. ~4 | ~7 | ~10 | ~19 | ~21 | 32 [avatar contradiction clause 356]**

**362. ~program(sK9) | ~program(sK9) <- (14, 15, 17) [resolution 146,209]**

**365. ~program(sK9) <- (14, 15, 17) [duplicate literal removal 362]**

**367. $false <- (14, 15, 17, 18) [subsumption resolution 365,159]**

**368. ~14 | ~15 | ~17 | ~18 [avatar contradiction clause 367]**

**370. ~program(sK4) | outputs(sK4,bad) | ~program(sK7(sK4)) <- (2, 6) [resolution 109,92]**

**371. ~program(sK9) | sP2(sK9,sK12(sK9)) | ~outputs(sK9,bad) | ~program(sK12(sK9)) <- (14, 17, 27) [resolution 198,209]**

**373. sP2(sK9,sK12(sK9)) | ~outputs(sK9,bad) | ~program(sK12(sK9)) <- (14, 17, 18, 27) [subsumption resolution 371,159]**

**376. 39 <=> program(sK12(sK9)) [avatar definition]**

**377. program(sK12(sK9)) <- (39) [avatar component clause 376]**

**378. ~program(sK12(sK9)) <- (~39) [avatar component clause 376]**

**380. 40 <=> sP2(sK9,sK12(sK9)) [avatar definition]**

**381. ~sP2(sK9,sK12(sK9)) <- (~40) [avatar component clause 380]**

**382. sP2(sK9,sK12(sK9)) <- (40) [avatar component clause 380]**

**384. ~program(sK9) <- (29, ~39) [resolution 378,207]**

**385. $false <- (18, 29, ~39) [subsumption resolution 384,159]**

**386. ~18 | ~29 | 39 [avatar contradiction clause 385]**

**389. ~halts2(sK9,sK12(sK9)) | ~outputs(sK9,good) <- (40) [resolution 382,81]**

**392. ~halts2(sK9,sK12(sK9)) <- (16, 40) [subsumption resolution 389,150]**

**393. ~program(sK12(sK9)) <- (14, 16, 17, 40) [resolution 392,209]**

**394. $false <- (14, 16, 17, 39, 40) [subsumption resolution 393,377]**

**395. ~14 | ~16 | ~17 | ~39 | ~40 [avatar contradiction clause 394]**

**396. sP2(sK9,sK12(sK9)) | ~outputs(sK9,bad) <- (14, 17, 18, 27, 39) [subsumption resolution 373,377]**

**397. ~13 | 40 | ~14 | ~17 | ~18 | ~27 | ~39 [avatar split clause 396,376,197,157,153,141,380,136]**

**401. ~program(sK12(X2)) | ~program(X2) | sP2(X2,sK12(X2)) <- (12, 28) [resolution 134,202]**

**403. sP2(X2,sK12(X2)) | ~program(X2) <- (12, 28, 29) [subsumption resolution 401,207]**

**404. ~program(X0) | ~outputs(sK4,good) | ~sP0(sK4,X0) <- (34) [resolution 311,71]**

**406. ~program(X0) | ~sP0(sK4,X0) <- (33, 34) [subsumption resolution 404,267]**

**407. ~sP0(sK4,X0) <- (33, 34) [subsumption resolution 406,69]**

**411. ~program(sK9) <- (12, 28, 29, ~40) [resolution 403,381]**

**412. $false <- (12, 18, 28, 29, ~40) [subsumption resolution 411,159]**

**413. ~12 | ~18 | ~28 | ~29 | 40 [avatar contradiction clause 412]**

**416. ~program(sK11) | ~halts2(sK11,sK11) <- (25, 38) [resolution 189,348]**

**420. ~halts2(sK11,sK11) <- (25, 26, 38) [subsumption resolution 416,194]**

**421. $false <- (25, 26, 38) [subsumption resolution 420,348]**

**422. ~25 | ~26 | ~38 [avatar contradiction clause 421]**

**423. outputs(sK4,bad) | ~program(sK7(sK4)) <- (2, 4, 6) [subsumption resolution 370,101]**

**425. 41 <=> outputs(sK4,bad) [avatar definition]**

**426. ~outputs(sK4,bad) <- (~41) [avatar component clause 425]**

**427. outputs(sK4,bad) <- (41) [avatar component clause 425]**

**428. ~32 | 41 | ~2 | ~4 | ~6 [avatar split clause 423,108,99,91,425,261]**

**433. ~program(sK4) | sP0(sK4,sK10(sK4)) | ~outputs(sK4,bad) | ~program(sK10(sK4)) <- (19, 34) [resolution 163,311]**

**442. sP0(sK4,sK10(sK4)) | ~outputs(sK4,bad) | ~program(sK10(sK4)) <- (4, 19, 34) [subsumption resolution 433,101]**

**443. ~outputs(sK4,bad) | ~program(sK10(sK4)) <- (4, 19, 33, 34) [subsumption resolution 442,407]**

**444. ~program(sK10(sK4)) <- (4, 19, 33, 34, 41) [subsumption resolution 443,427]**

**445. ~program(sK4) <- (4, 19, 21, 33, 34, 41) [resolution 444,172]**

**446. $false <- (4, 19, 21, 33, 34, 41) [subsumption resolution 445,101]**

**447. ~4 | ~19 | ~21 | ~33 | ~34 | ~41 [avatar contradiction clause 446]**

**450. 42 <=> program(sK10(sK4)) [avatar definition]**

**451. program(sK10(sK4)) <- (42) [avatar component clause 450]**

**452. ~program(sK10(sK4)) <- (~42) [avatar component clause 450]**

**454. 43 <=> sP0(sK4,sK10(sK4)) [avatar definition]**

**456. sP0(sK4,sK10(sK4)) <- (43) [avatar component clause 454]**

**480. halts2(sK9,X1) | ~program(X1) <- (8, 14) [subsumption resolution 142,117]**

**481. ~program(X0) | ~program(sK9) <- (8, 14) [resolution 480,117]**

**483. ~program(X0) <- (8, 14, 18) [subsumption resolution 481,159]**

**488. $false <- (8, 14, 18) [resolution 483,159]**

**495. ~8 | ~14 | ~18 [avatar contradiction clause 488]**

**507. ~program(sK4) <- (21, ~42) [resolution 452,172]**

**508. $false <- (4, 21, ~42) [subsumption resolution 507,101]**

**509. ~4 | ~21 | 42 [avatar contradiction clause 508]**

**511. halts2(sK10(sK4),sK10(sK4)) <- (43) [resolution 456,70]**

**514. ~program(sK10(sK4)) <- (8, 43) [resolution 511,117]**

**515. $false <- (8, 42, 43) [subsumption resolution 514,451]**

**516. ~8 | ~42 | ~43 [avatar contradiction clause 515]**

**519. sP0(sK4,sK10(sK4)) | ~outputs(sK4,bad) <- (4, 19, 34, 42) [subsumption resolution 442,451]**

**520. ~41 | 43 | ~4 | ~19 | ~34 | ~42 [avatar split clause 519,450,310,162,99,454,425]**

**521. ~program(sK4) | program(sK7(sK4)) <- (7, ~41) [resolution 426,113]**

**522. program(sK7(sK4)) <- (4, 7, ~41) [subsumption resolution 521,101]**

**523. $false <- (4, 7, ~32, ~41) [subsumption resolution 522,263]**

**524. ~4 | ~7 | 32 | 41 [avatar contradiction clause 523]**

**525. $false [avatar sat refutation 93,97,102,103,110,114,121,125,139,143,151,155,160,164,173,186,190,195,199,203,208,217,219,307,326,349,351,353,357,368,386,395,397,413,422,428,447,495,509,516,520,524]**

**% SZS output end Proof for COM003+1**

**% ------------------------------**

**% Version: Vampire 4.5.1 (commit )**

**% Termination reason: Refutation**

**% Memory used [KB]: 5117**

**% Time elapsed: 0.001 s**

**% ------------------------------**

**% ------------------------------**

**Summary of proof**

Although the proof in this Appendix may seem formidable, it has a high-level intelligible structure:

Steps 1-4 assert axioms p1 – p4.

Step 5 asserts the Halting Theorem.

Step 6 is the negation of the Halting Theorem.

Steps 7-11 partially orthogonalize and standardize (a translation VAMPIRE calls “rectifying”) the naming of the variables occurring in Steps 1-6.

Step 12 removes the double negation at the front end of Step 11 (this is one of several translations VAMPIRE calls “flattening”.)

Steps 13, 14, 16, 18, and 20 replace the symbol “=>” with logical equivalents, i.e., replace expressions of the form X => Y with ~X | Y.

Steps 15, 17, and 19 flatten the clauses indicated.

Steps 21-26 rename/consolidate some recurring subclauses.

Steps 27-51 rectify and skolemize (see Chang and Keisler 2012, Chap. 3) various sentences.

Steps 52-86 transform various sentences to Clause Normal Form (cnf), a form that is required by some of VAMPIRE’s inference rules.

Steps 88-267 rename various clauses and subclauses.

Steps 269-306, 307-326, 336-357, 362-368, 370-386, 389-395, 396-413, 416-422, 423-447, 450-495, 507-509, 511-516, and 519-524 collectively show that there is a partitioning of the proof space such that each inference chain in that space yields a contradiction.

Step 525 says that all the chains in the partition mentioned in Step 524 yield a contradiction. Thus, by proof by contradiction, Step 6 is false, i.e., the Halting Theorem follows from p1 – p4.

**APPENDIX 2. Proof of independence of p1.** **See Sutcliffe 2008; Sutcliffe, Zimmer, and Schulz 2003, 2004; Sutcliffe and Suttner 2021 for syntax and ontology details.**

**Command line:**

$ ../bin/vampire\_rel.exe –saturation\_algorithm fmb *input\_file* > *output\_file*

**VAMPIRE input file (negates premise p1).**

**%--------------------------------------------------------------------------**

**fof(p1,axiom,**

**( ~ (? [X] :**

**( algorithm(X)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(X,Y,Z) ) )**

**=> ? [W] :**

**( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) ) ))).**

**fof(p2,axiom,**

**( ! [W] :**

**( ( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) )**

**=> ! [Y,Z] :**

**( ( ( program(Y)**

**& halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,bad) ) ) ) ) )).**

**fof(p3,axiom,**

**( ? [W] :**

**( program(W)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,bad) ) ) ) )**

**=> ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) ) )).**

**fof(p4,axiom,**

**( ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) )**

**=> ? [U] :**

**( program(U)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ~ halts2(U,Y) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(U,Y)**

**& outputs(U,bad) ) ) ) ) )).**

**%--------------------------------------------------------------------------**

**VAMPIRE output file (model showing independence of p1)**

**WARNING! Could not set resource limit: Virtual memory.**

**TRYING [1]**

**Finite Model Found!**

**% SZS status Satisfiable for COM003\_negp1**

**% SZS output start FiniteModel for COM003\_negp1**

**tff(declare\_$i,type,$i:$tType).**

**tff(declare\_$i1,type,good:$i).**

**tff(finite\_domain,axiom,**

**! [X:$i] : (**

**X = good**

**) ).**

**tff(declare\_bad,type,bad:$i).**

**tff(bad\_definition,axiom,bad = good).**

**tff(declare\_algorithm,type,algorithm: $i > $o ).**

**tff(predicate\_algorithm,axiom,**

**% algorithm(good) undefined in model**

**).**

**tff(declare\_program,type,program: $i > $o ).**

**tff(predicate\_program,axiom,**

**~program(good)**

**).**

**tff(declare\_decides,type,decides: $i \* $i \* $i > $o ).**

**tff(predicate\_decides,axiom,**

**~decides(good,good,good)**

**).**

**tff(declare\_halts2,type,halts2: $i \* $i > $o ).**

**tff(predicate\_halts2,axiom,**

**~halts2(good,good)**

**).**

**tff(declare\_halts3,type,halts3: $i \* $i \* $i > $o ).**

**tff(predicate\_halts3,axiom,**

**~halts3(good,good,good)**

**).**

**tff(declare\_outputs,type,outputs: $i \* $i > $o ).**

**tff(predicate\_outputs,axiom,**

**outputs(good,good)**

**).**

**% SZS output end FiniteModel for COM003\_negp1**

**% ------------------------------**

**% Version: Vampire 4.5.1 (commit )**

**% Termination reason: Satisfiable**

**% Memory used [KB]: 4861**

**% Time elapsed: 0.001 s**

**% ------------------------------**

**% ------------------------------**

**APPENDIX 3. Proof of independence of p2. See Sutcliffe 2008; Sutcliffe, Zimmer, and Schulz 2003, 2004; Sutcliffe and Suttner 2021 for syntax and ontology details.**

**Command line:**

$ ../bin/vampire\_rel.exe –saturation\_algorithm fmb *input\_file* > *output\_file*

**VAMPIRE input file (negates premise p2).**

**%--------------------------------------------------------------------------**

**fof(p1,axiom,**

**( ? [X] :**

**( algorithm(X)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(X,Y,Z) ) )**

**=> ? [W] :**

**( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) ) )).**

**fof(p2,axiom,**

**( ~ ( ! [W] :**

**( ( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) )**

**=> ! [Y,Z] :**

**( ( ( program(Y)**

**& halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,bad) ) ) ) ) ))).**

**fof(p3,axiom,**

**( ? [W] :**

**( program(W)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,bad) ) ) ) )**

**=> ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) ) )).**

**fof(p4,axiom,**

**( ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) )**

**=> ? [U] :**

**( program(U)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ~ halts2(U,Y) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(U,Y)**

**& outputs(U,bad) ) ) ) ) )).**

**%--------------------------------------------------------------------------**

**VAMPIRE output file (model showing independence of p2)**

**WARNING! Could not set resource limit: Virtual memory.**

**TRYING [1]**

**Finite Model Found!**

**% SZS status Satisfiable for COM003\_negp2**

**% SZS output start FiniteModel for COM003\_negp2**

**tff(declare\_$i,type,$i:$tType).**

**tff(declare\_$i1,type,good:$i).**

**tff(finite\_domain,axiom,**

**! [X:$i] : (**

**X = good**

**) ).**

**tff(declare\_bad,type,bad:$i).**

**tff(bad\_definition,axiom,bad = good).**

**tff(declare\_algorithm,type,algorithm: $i > $o ).**

**tff(predicate\_algorithm,axiom,**

**% algorithm(good) undefined in model**

**).**

**tff(declare\_program,type,program: $i > $o ).**

**tff(predicate\_program,axiom,**

**program(good)**

**).**

**tff(declare\_decides,type,decides: $i \* $i \* $i > $o ).**

**tff(predicate\_decides,axiom,**

**% decides(good,good,good) undefined in model**

**).**

**tff(declare\_halts2,type,halts2: $i \* $i > $o ).**

**tff(predicate\_halts2,axiom,**

**~halts2(good,good)**

**).**

**tff(declare\_halts3,type,halts3: $i \* $i \* $i > $o ).**

**tff(predicate\_halts3,axiom,**

**~halts3(good,good,good)**

**).**

**tff(declare\_outputs,type,outputs: $i \* $i > $o ).**

**tff(predicate\_outputs,axiom,**

**~outputs(good,good)**

**).**

**% SZS output end FiniteModel for COM003\_negp2**

**% ------------------------------**

**% Version: Vampire 4.5.1 (commit )**

**% Termination reason: Satisfiable**

**% Memory used [KB]: 4861**

**% Time elapsed: 0.140 s**

**% ------------------------------**

**% ------------------------------**

**APPENDIX 4. Proof of independence of p3. See Sutcliffe 2008; Sutcliffe, Zimmer, and Schulz 2003, 2004; Sutcliffe and Suttner 2021 for syntax and ontology details.**

**Command line:**

$ ../bin/vampire\_rel.exe –saturation\_algorithm fmb *input\_file* > *output\_file*

**VAMPIRE input file (negates premise p3).**

**%--------------------------------------------------------------------------**

**fof(p1,axiom,**

**( ? [X] :**

**( algorithm(X)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(X,Y,Z) ) )**

**=> ? [W] :**

**( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) ) )).**

**fof(p2,axiom,**

**( ! [W] :**

**( ( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) )**

**=> ! [Y,Z] :**

**( ( ( program(Y)**

**& halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,bad) ) ) ) ) )).**

**fof(p3,axiom,**

**( ~ ( ? [W] :**

**( program(W)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,bad) ) ) ) )**

**=> ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) ) ))).**

**fof(p4,axiom,**

**( ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) )**

**=> ? [U] :**

**( program(U)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ~ halts2(U,Y) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(U,Y)**

**& outputs(U,bad) ) ) ) ) )).**

**%--------------------------------------------------------------------------**

**VAMPIRE output file (model showing independence of p3)**

**%--------------------------------------------------------------------------**

**fof(p1,axiom,**

**( ? [X] :**

**( algorithm(X)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(X,Y,Z) ) )**

**=> ? [W] :**

**( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) ) )).**

**fof(p2,axiom,**

**( ! [W] :**

**( ( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) )**

**=> ! [Y,Z] :**

**( ( ( program(Y)**

**& halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,bad) ) ) ) ) )).**

**fof(p3,axiom,**

**( ~ ( ? [W] :**

**( program(W)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,bad) ) ) ) )**

**=> ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) ) ))).**

**fof(p4,axiom,**

**( ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) )**

**=> ? [U] :**

**( program(U)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ~ halts2(U,Y) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(U,Y)**

**& outputs(U,bad) ) ) ) ) )).**

**%--------------------------------------------------------------------------**

**APPENDIX 5. Proof of independence of p4. See Sutcliffe 2008; Sutcliffe, Zimmer, and Schulz 2003, 2004; Sutcliffe and Suttner 2021 for syntax and ontology details.**

**Command line:**

$ ../bin/vampire\_rel.exe –saturation\_algorithm fmb *input\_file* > *output\_file*

**VAMPIRE input file (negates premise p4).**

**%--------------------------------------------------------------------------**

**fof(p1,axiom,**

**( ? [X] :**

**( algorithm(X)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(X,Y,Z) ) )**

**=> ? [W] :**

**( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) ) )).**

**fof(p2,axiom,**

**( ! [W] :**

**( ( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) )**

**=> ! [Y,Z] :**

**( ( ( program(Y)**

**& halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,bad) ) ) ) ) )).**

**fof(p3,axiom,**

**( ? [W] :**

**( program(W)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,bad) ) ) ) )**

**=> ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) ) )).**

**fof(p4,axiom,**

**( ~ ( ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) )**

**=> ? [U] :**

**( program(U)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ~ halts2(U,Y) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(U,Y)**

**& outputs(U,bad) ) ) ) ) ))).**

**%--------------------------------------------------------------------------**

**VAMPIRE output file (model showing independence of p4)**

**WARNING! Could not set resource limit: Virtual memory.**

**TRYING [1]**

**Finite Model Found!**

**% SZS status Satisfiable for COM003\_negp4**

**% SZS output start FiniteModel for COM003\_negp4**

**tff(declare\_$i,type,$i:$tType).**

**tff(declare\_$i1,type,good:$i).**

**tff(finite\_domain,axiom,**

**! [X:$i] : (**

**X = good**

**) ).**

**tff(declare\_bad,type,bad:$i).**

**tff(bad\_definition,axiom,bad = good).**

**tff(declare\_algorithm,type,algorithm: $i > $o ).**

**tff(predicate\_algorithm,axiom,**

**% algorithm(good) undefined in model**

**).**

**tff(declare\_program,type,program: $i > $o ).**

**tff(predicate\_program,axiom,**

**program(good)**

**).**

**tff(declare\_decides,type,decides: $i \* $i \* $i > $o ).**

**tff(predicate\_decides,axiom,**

**% decides(good,good,good) undefined in model**

**).**

**tff(declare\_halts2,type,halts2: $i \* $i > $o ).**

**tff(predicate\_halts2,axiom,**

**halts2(good,good)**

**).**

**tff(declare\_halts3,type,halts3: $i \* $i \* $i > $o ).**

**tff(predicate\_halts3,axiom,**

**~halts3(good,good,good)**

**).**

**tff(declare\_outputs,type,outputs: $i \* $i > $o ).**

**tff(predicate\_outputs,axiom,**

**outputs(good,good)**

**).**

**% SZS output end FiniteModel for COM003\_negp4**

**% ------------------------------**

**% Version: Vampire 4.5.1 (commit )**

**% Termination reason: Satisfiable**

**% Memory used [KB]: 4861**

**% Time elapsed: 0.031 s**

**% ------------------------------**

**% ------------------------------**

**APPENDIX 6. Proof of consistency of Burkholder 1987 axioms. See Sutcliffe 2008; Sutcliffe, Zimmer, and Schulz 2003, 2004; Sutcliffe and Suttner 2021 for syntax and ontology details.**

**Command line:**

$ ../bin/vampire\_rel.exe –saturation\_algorithm fmb *input\_file* > *output\_file*

**VAMPIRE input file (same as Figure 3).**

**%--------------------------------------------------------------------------**

**fof(p1,axiom,**

**( ? [X] :**

**( algorithm(X)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(X,Y,Z) ) )**

**=> ? [W] :**

**( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) ) )).**

**fof(p2,axiom,**

**( ! [W] :**

**( ( program(W)**

**& ! [Y] :**

**( program(Y)**

**=> ! [Z] : decides(W,Y,Z) ) )**

**=> ! [Y,Z] :**

**( ( ( program(Y)**

**& halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Z) )**

**=> ( halts3(W,Y,Z)**

**& outputs(W,bad) ) ) ) ) )).**

**fof(p3,axiom,**

**( ? [W] :**

**( program(W)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts3(W,Y,Y)**

**& outputs(W,bad) ) ) ) )**

**=> ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) ) )).**

**fof(p4,axiom,**

**( ? [V] :**

**( program(V)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,good) ) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(V,Y)**

**& outputs(V,bad) ) ) ) )**

**=> ? [U] :**

**( program(U)**

**& ! [Y] :**

**( ( ( program(Y)**

**& halts2(Y,Y) )**

**=> ~ halts2(U,Y) )**

**& ( ( program(Y)**

**& ~ halts2(Y,Y) )**

**=> ( halts2(U,Y)**

**& outputs(U,bad) ) ) ) ) )).**

**%--------------------------------------------------------------------------**

**VAMPIRE output file (model showing consistency of Burkholder 1987 axioms)**

**WARNING! Could not set resource limit: Virtual memory.**

**TRYING [1]**

**Finite Model Found!**

**% SZS status Satisfiable for COM003**

**% SZS output start FiniteModel for COM003**

**tff(declare\_$i,type,$i:$tType).**

**tff(declare\_$i1,type,good:$i).**

**tff(finite\_domain,axiom,**

**! [X:$i] : (**

**X = good**

**) ).**

**tff(declare\_bad,type,bad:$i).**

**tff(bad\_definition,axiom,bad = good).**

**tff(declare\_algorithm,type,algorithm: $i > $o ).**

**tff(predicate\_algorithm,axiom,**

**% algorithm(good) undefined in model**

**).**

**tff(declare\_program,type,program: $i > $o ).**

**tff(predicate\_program,axiom,**

**~program(good)**

**).**

**tff(declare\_decides,type,decides: $i \* $i \* $i > $o ).**

**tff(predicate\_decides,axiom,**

**% decides(good,good,good) undefined in model**

**).**

**tff(declare\_halts2,type,halts2: $i \* $i > $o ).**

**tff(predicate\_halts2,axiom,**

**halts2(good,good)**

**).**

**tff(declare\_halts3,type,halts3: $i \* $i \* $i > $o ).**

**tff(predicate\_halts3,axiom,**

**~halts3(good,good,good)**

**).**

**tff(declare\_outputs,type,outputs: $i \* $i > $o ).**

**tff(predicate\_outputs,axiom,**

**~outputs(good,good)**

**).**

**% SZS output end FiniteModel for COM003**

**% ------------------------------**

**% Version: Vampire 4.5.1 (commit )**

**% Termination reason: Satisfiable**

**% Memory used [KB]: 4861**

**% Time elapsed: 0.041 s**

**% ------------------------------**

**% ------------------------------**