SWW axioms
SWW problems

SWW096+1.p Equivalenace of the semantic and syntactic definition of and
include('Axioms/SWV012+0.ax')
$\forall p, q$: $\text{and}_1(p, q) = \text{and}_2(p, q)$ fof(and1_and$_2$, conjecture)

SWW097+1.p Equivalenace of the semantic and syntactic definition of lazy_and
include('Axioms/SWV012+0.ax')
$\forall p, q$: $\text{lazy\_and}_1(p, q) = \text{lazy\_and}_2(p, q)$ fof(lazy_and1_lazy_and$_2$, conjecture)

SWW098+1.p Equivalence of or1 and or2
include('Axioms/SWV012+0.ax')
$\forall p, q$: $\text{or}_1(p, q) = \text{or}_2(p, q)$ fof(or1_or$_2$, conjecture)

SWW099+1.p If one is Boolean then exists1(P) = exists2(P).
include('Axioms/SWV012+0.ax')
$\forall p$: $((\text{bool}(\text{exists}_1(p)) \text{ or } \text{bool}(\text{exists}_2(p))) \Rightarrow \text{exists}_1(p) = \text{exists}_2(p))$ fof(exists1_exists$_2$, conjecture)

SWW100+1.p If only one element non-Boolean, then exists1(P) = exists2(P)
include('Axioms/SWV012+0.ax')
$\forall p$: $(\forall x_1, x_2$: $((\neg \text{bool}(\text{apply}(p, x_1)) \text{ and } \neg \text{bool}(\text{apply}(p, x_2))) \Rightarrow \text{apply}(p, x_1) = \text{apply}(p, x_2)) \Rightarrow \text{exists}_1(p) = \text{exists}_2(p))$ fof(exists1_exists$_2$, conjecture)

SWW101+1.p false1 = false2
include('Axioms/SWV012+0.ax')
$\text{false}_1 = \text{false}_2$ fof(false1_false$_2$, conjecture)

SWW102+1.p Equivalence of not1 and not2
include('Axioms/SWV012+0.ax')
$\forall p$: $\text{not}_1(p) = \text{not}_2(p)$ fof(not1_not$_2$, conjecture)

SWW103+1.p Syntactic definitions of the logical operators
include('Axioms/SWV012+0.ax')

SWW397-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise$_1$, hypothesis)
$\text{heap}(\text{sep}(\text{lseg}(x_2, \text{nil}), \text{emp}))$ cnf(premise$_2$, hypothesis)
$x_2 = x_2 \Rightarrow \neg \text{heap}(\text{emp})$ cnf(conclusion$_1$, negated_conjecture)

SWW398-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise$_1$, hypothesis)
$\text{heap}(\text{sep}(\text{next}(x_1, \text{nil}), \text{emp}))$ cnf(premise$_2$, hypothesis)
$\neg \text{heap}(\text{sep}(\text{lseg}(\text{nil}, \text{nil}), \text{emp}))$ cnf(conclusion$_1$, negated_conjecture)

SWW399-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise$_1$, hypothesis)
$\text{heap}(\text{sep}(\text{next}(x_1, \text{nil}), \text{emp}))$ cnf(premise$_2$, hypothesis)
$\neg \text{heap}(\text{sep}(\text{lseg}(x_1, \text{nil}), \text{emp}))$ cnf(conclusion$_1$, negated_conjecture)

SWW400-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise$_1$, hypothesis)

$\mathrm{nil} \neq x_2$ cnf(premise$_2$, hypothesis)
$\mathrm{heap}(\mathrm{sep}(\mathrm{next}(x_2, \mathrm{nil}), \mathrm{sep}(\mathrm{lseg}(x_1, x_2), \mathrm{emp})))$ cnf(premise$_3$, hypothesis)
$\neg\,\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_1, \mathrm{nil}), \mathrm{emp}))$ cnf(conclusion$_1$, negated_conjecture)

SWW401-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\mathrm{nil} \neq x_1$ cnf(premise$_1$, hypothesis)
$\mathrm{nil} \neq x_2$ cnf(premise$_2$, hypothesis)
$\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_1, x_2), \mathrm{sep}(\mathrm{next}(x_2, x_1), \mathrm{emp})))$ cnf(premise$_3$, hypothesis)
$\neg\,\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_3, x_2), \mathrm{sep}(\mathrm{next}(x_2, x_3), \mathrm{emp})))$ cnf(conclusion$_1$, negated_conjecture)

SWW402-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\mathrm{nil} \neq x_1$ cnf(premise$_1$, hypothesis)
$\mathrm{nil} \neq x_2$ cnf(premise$_2$, hypothesis)
$\mathrm{nil} \neq x_3$ cnf(premise$_3$, hypothesis)
$\mathrm{nil} \neq x_4$ cnf(premise$_4$, hypothesis)
$\mathrm{nil} \neq x_5$ cnf(premise$_5$, hypothesis)
$\mathrm{nil} \neq x_6$ cnf(premise$_6$, hypothesis)
$x_1 \neq x_6$ cnf(premise$_7$, hypothesis)
$x_2 \neq x_6$ cnf(premise$_8$, hypothesis)
$x_3 \neq x_4$ cnf(premise$_9$, hypothesis)
$x_3 \neq x_5$ cnf(premise$_{10}$, hypothesis)
$\mathrm{heap}(\mathrm{sep}(\mathrm{next}(x_1, x_6), \mathrm{sep}(\mathrm{lseg}(x_2, x_1), \mathrm{sep}(\mathrm{next}(x_6, x_2), \mathrm{emp}))))$ cnf(premise$_{11}$, hypothesis)
$\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_7, x_6), \mathrm{sep}(\mathrm{next}(x_6, x_7), \mathrm{emp}))) \Rightarrow x_7 = x_6$ cnf(conclusion$_1$, negated_conjecture)

SWW403-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\mathrm{nil} \neq x_1$ cnf(premise$_1$, hypothesis)
$\mathrm{nil} \neq x_2$ cnf(premise$_2$, hypothesis)
$\mathrm{nil} \neq x_3$ cnf(premise$_3$, hypothesis)
$\mathrm{nil} \neq x_4$ cnf(premise$_4$, hypothesis)
$x_2 \neq x_3$ cnf(premise$_5$, hypothesis)
$x_2 \neq x_4$ cnf(premise$_6$, hypothesis)
$\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_1, x_4), \mathrm{sep}(\mathrm{next}(x_4, x_1), \mathrm{emp})))$ cnf(premise$_7$, hypothesis)
$\neg\,\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_5, x_4), \mathrm{sep}(\mathrm{next}(x_4, x_5), \mathrm{emp})))$ cnf(conclusion$_1$, negated_conjecture)

SWW404-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\mathrm{nil} \neq x_1$ cnf(premise$_1$, hypothesis)
$\mathrm{nil} \neq x_2$ cnf(premise$_2$, hypothesis)
$\mathrm{nil} \neq x_3$ cnf(premise$_3$, hypothesis)
$x_1 \neq x_2$ cnf(premise$_4$, hypothesis)
$x_1 \neq x_3$ cnf(premise$_5$, hypothesis)
$\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_2, x_3), \mathrm{sep}(\mathrm{next}(x_1, x_2), \mathrm{sep}(\mathrm{next}(x_3, x_1), \mathrm{emp}))))$ cnf(premise$_6$, hypothesis)
$\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_4, x_3), \mathrm{sep}(\mathrm{next}(x_3, x_4), \mathrm{emp}))) \Rightarrow x_4 = x_3$ cnf(conclusion$_1$, negated_conjecture)

SWW405-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\mathrm{nil} \neq x_1$ cnf(premise$_1$, hypothesis)
$\mathrm{nil} \neq x_2$ cnf(premise$_2$, hypothesis)
$\mathrm{nil} \neq x_3$ cnf(premise$_3$, hypothesis)

$x_1 \neq x_2$ cnf(premise$_4$, hypothesis)
$x_2 \neq x_3$ cnf(premise$_5$, hypothesis)
heap(sep(lseg(x_3, x_1), sep(next(x_1, x_3), emp))) cnf(premise$_6$, hypothesis)
$\neg$ heap(sep(lseg(x_4, x_1), sep(next(x_1, x_4), emp))) cnf(conclusion$_1$, negated_conjecture)

SWW406-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
nil $\neq x_2$ cnf(premise$_2$, hypothesis)
nil $\neq x_3$ cnf(premise$_3$, hypothesis)
$x_1 \neq x_3$ cnf(premise$_4$, hypothesis)
$x_2 \neq x_3$ cnf(premise$_5$, hypothesis)
heap(sep(lseg(x_2, x_1), sep(next(x_3, x_2), sep(next(x_1, x_3), emp)))) cnf(premise$_6$, hypothesis)
$\neg$ heap(sep(next(x_4, x_3), sep(lseg(x_2, x_4), sep(next(x_3, x_2), emp)))) cnf(conclusion$_1$, negated_conjecture)

SWW407-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
nil $\neq x_2$ cnf(premise$_2$, hypothesis)
nil $\neq x_3$ cnf(premise$_3$, hypothesis)
$x_1 \neq x_4$ cnf(premise$_4$, hypothesis)
$x_1 \neq x_2$ cnf(premise$_5$, hypothesis)
$x_4 \neq x_2$ cnf(premise$_6$, hypothesis)
$x_4 \neq x_3$ cnf(premise$_7$, hypothesis)
$x_2 \neq x_3$ cnf(premise$_8$, hypothesis)
heap(sep(next(x_1, x_2), sep(lseg(x_3, x_1), sep(lseg(x_4, nil), sep(next(x_2, x_4), emp))))) cnf(premise$_9$, hypothesis)
$\neg$ heap(sep(lseg(x_4, nil), sep(next(x_2, x_4), sep(lseg(x_3, x_2), emp)))) cnf(conclusion$_1$, negated_conjecture)

SWW408-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
nil $\neq x_2$ cnf(premise$_2$, hypothesis)
$x_1 \neq x_2$ cnf(premise$_3$, hypothesis)
$x_1 \neq x_3$ cnf(premise$_4$, hypothesis)
$x_2 \neq x_3$ cnf(premise$_5$, hypothesis)
heap(sep(next(x_1, x_2), sep(lseg(x_3, nil), sep(next(x_2, x_3), emp)))) cnf(premise$_6$, hypothesis)
$\neg$ heap(sep(lseg(x_3, nil), emp)) cnf(conclusion$_1$, negated_conjecture)

SWW409-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
nil $\neq x_2$ cnf(premise$_2$, hypothesis)
$x_1 \neq x_3$ cnf(premise$_3$, hypothesis)
$x_2 \neq x_3$ cnf(premise$_4$, hypothesis)
heap(sep(lseg(x_2, x_1), sep(lseg(x_3, nil), sep(next(x_1, x_3), emp)))) cnf(premise$_5$, hypothesis)
$\neg$ heap(sep(lseg(x_2, nil), emp)) cnf(conclusion$_1$, negated_conjecture)

SWW410-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
nil $\neq x_2$ cnf(premise$_2$, hypothesis)
$x_1 \neq x_3$ cnf(premise$_3$, hypothesis)

$x_3 \neq x_2$ cnf(premise$_4$, hypothesis)
heap(sep(lseg(x_2, x_1), sep(lseg(x_3, nil), sep(next(x_1, x_3), emp)))) cnf(premise$_5$, hypothesis)
$\neg$ heap(sep(lseg(x_3, nil), sep(lseg(x_2, x_3), emp))) cnf(conclusion$_1$, negated_conjecture)

SWW411-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
nil $\neq x_2$ cnf(premise$_2$, hypothesis)
$x_3 \neq x_1$ cnf(premise$_3$, hypothesis)
$x_3 \neq x_2$ cnf(premise$_4$, hypothesis)
$x_1 \neq x_2$ cnf(premise$_5$, hypothesis)
heap(sep(lseg(x_1, nil), sep(lseg(x_3, nil), sep(next(x_2, x_3), emp)))) cnf(premise$_6$, hypothesis)
$\neg$ heap(sep(lseg(x_1, nil), sep(lseg(x_3, nil), emp))) cnf(conclusion$_1$, negated_conjecture)

SWW412-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
nil $\neq x_2$ cnf(premise$_2$, hypothesis)
$x_3 \neq x_1$ cnf(premise$_3$, hypothesis)
$x_3 \neq x_2$ cnf(premise$_4$, hypothesis)
$x_1 \neq x_2$ cnf(premise$_5$, hypothesis)
$x_1 \neq x_4$ cnf(premise$_6$, hypothesis)
$x_2 \neq x_4$ cnf(premise$_7$, hypothesis)
heap(sep(lseg(x_3, nil), sep(lseg(x_4, nil), sep(next(x_1, x_4), sep(next(x_2, x_3), emp))))) cnf(premise$_8$, hypothesis)
$\neg$ heap(sep(lseg(x_2, nil), sep(lseg(x_1, nil), emp))) cnf(conclusion$_1$, negated_conjecture)

SWW413-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
nil $\neq x_2$ cnf(premise$_2$, hypothesis)
$x_3 \neq x_2$ cnf(premise$_3$, hypothesis)
$x_4 \neq x_1$ cnf(premise$_4$, hypothesis)
$x_4 \neq x_2$ cnf(premise$_5$, hypothesis)
$x_1 \neq x_2$ cnf(premise$_6$, hypothesis)
heap(sep(lseg(x_3, nil), sep(next(x_2, x_3), emp))) cnf(premise$_7$, hypothesis)
$\neg$ heap(sep(lseg(x_2, nil), emp)) cnf(conclusion$_1$, negated_conjecture)

SWW414-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
$x_1 \neq x_2$ cnf(premise$_2$, hypothesis)
heap(sep(lseg(x_2, nil), sep(lseg(x_1, nil), emp))) cnf(premise$_3$, hypothesis)
$\neg$ heap(sep(lseg(x_2, nil), emp)) cnf(conclusion$_1$, negated_conjecture)

SWW415-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
$x_1 \neq x_2$ cnf(premise$_2$, hypothesis)
heap(sep(lseg(x_3, nil), emp)) cnf(premise$_3$, hypothesis)
$x_3 = x_3 \Rightarrow \neg$ heap(emp) cnf(conclusion$_1$, negated_conjecture)

SWW416-1.p Verification Condition generated by Smallfoot

This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise_1, hypothesis)
$x_1 \neq x_2$ cnf(premise_2, hypothesis)
$x_1 \neq x_3$ cnf(premise_3, hypothesis)
$\text{heap}(\text{sep}(\text{lseg}(x_3, \text{nil}), \text{sep}(\text{lseg}(x_1, \text{nil}), \text{emp})))$ cnf(premise_4, hypothesis)
$\neg\, \text{heap}(\text{sep}(\text{lseg}(x_1, \text{nil}), \text{emp}))$ cnf(conclusion_1, negated_conjecture)

SWW417-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise_1, hypothesis)
$x_2 \neq x_1$ cnf(premise_2, hypothesis)
$\text{heap}(\text{sep}(\text{lseg}(x_2, \text{nil}), \text{sep}(\text{next}(x_1, x_2), \text{emp})))$ cnf(premise_3, hypothesis)
$\neg\, \text{heap}(\text{sep}(\text{lseg}(x_1, \text{nil}), \text{emp}))$ cnf(conclusion_1, negated_conjecture)

SWW418-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise_1, hypothesis)
$x_2 \neq x_1$ cnf(premise_2, hypothesis)
$\text{heap}(\text{sep}(\text{lseg}(x_2, \text{nil}), \text{sep}(\text{next}(x_1, x_2), \text{emp})))$ cnf(premise_3, hypothesis)
$\neg\, \text{heap}(\text{sep}(\text{lseg}(x_2, \text{nil}), \text{emp}))$ cnf(conclusion_1, negated_conjecture)

SWW419-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise_1, hypothesis)
$x_2 \neq x_1$ cnf(premise_2, hypothesis)
$x_1 \neq x_3$ cnf(premise_3, hypothesis)
$\text{heap}(\text{sep}(\text{next}(x_1, x_3), \text{sep}(\text{lseg}(x_2, \text{nil}), \text{emp})))$ cnf(premise_4, hypothesis)
$x_2 = x_2 \Rightarrow \neg\, \text{heap}(\text{emp})$ cnf(conclusion_1, negated_conjecture)

SWW420-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise_1, hypothesis)
$x_2 \neq x_1$ cnf(premise_2, hypothesis)
$x_1 \neq x_3$ cnf(premise_3, hypothesis)
$\text{heap}(\text{sep}(\text{lseg}(x_3, \text{nil}), \text{sep}(\text{lseg}(x_2, \text{nil}), \text{sep}(\text{next}(x_1, x_2), \text{emp}))))$ cnf(premise_4, hypothesis)
$\neg\, \text{heap}(\text{sep}(\text{lseg}(x_2, \text{nil}), \text{sep}(\text{next}(x_1, x_2), \text{sep}(\text{lseg}(x_1, x_1), \text{emp}))))$ cnf(conclusion_1, negated_conjecture)

SWW421-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise_1, hypothesis)
$x_2 \neq x_1$ cnf(premise_2, hypothesis)
$x_1 \neq x_3$ cnf(premise_3, hypothesis)
$\text{heap}(\text{sep}(\text{lseg}(x_2, x_3), \text{sep}(\text{next}(x_1, x_2), \text{emp})))$ cnf(premise_4, hypothesis)
$\neg\, \text{heap}(\text{sep}(\text{lseg}(x_1, x_3), \text{emp}))$ cnf(conclusion_1, negated_conjecture)

SWW422-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
$\text{nil} \neq x_1$ cnf(premise_1, hypothesis)
$x_2 \neq x_1$ cnf(premise_2, hypothesis)

$x_1 \neq x_3$ cnf(premise$_3$, hypothesis)
heap(sep(lseg(x_2, x_3), sep(next(x_1, x_2), emp))) cnf(premise$_4$, hypothesis)
$x_3 = x_3 \Rightarrow \neg$ heap(sep(lseg(x_2, x_3), emp)) cnf(conclusion$_1$, negated_conjecture)

SWW423-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
$x_2 \neq x_1$ cnf(premise$_2$, hypothesis)
$x_1 \neq x_3$ cnf(premise$_3$, hypothesis)
heap(sep(lseg(x_2, nil), sep(lseg(x_3, nil), sep(next(x_1, x_2), emp)))) cnf(premise$_4$, hypothesis)
$\neg$ heap(sep(lseg(x_1, nil), sep(lseg(x_3, nil), emp))) cnf(conclusion$_1$, negated_conjecture)

SWW424-1.p Verification Condition generated by Smallfoot
This is one of the verification conditions that were gathered from the output of Smallfoot when checking assertions on list manipulating programs from its own benchmark suite.
include('Axioms/SWV013-0.ax')
nil $\neq x_1$ cnf(premise$_1$, hypothesis)
$x_2 \neq x_1$ cnf(premise$_2$, hypothesis)
$x_2 \neq x_3$ cnf(premise$_3$, hypothesis)
$x_1 \neq x_3$ cnf(premise$_4$, hypothesis)
heap(sep(next(x_1, x_3), sep(lseg(x_2, nil), emp))) cnf(premise$_5$, hypothesis)
$\neg$ heap(sep(lseg(x_2, nil), emp)) cnf(conclusion$_1$, negated_conjecture)

SWW425-1.p Randomly generated entailment of the form F → ⊥ (n = 10)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_1 \neq x_6$ cnf(premise$_1$, hypothesis)
$x_1 \neq x_7$ cnf(premise$_2$, hypothesis)
$x_1 \neq x_{10}$ cnf(premise$_3$, hypothesis)
$x_4 \neq x_8$ cnf(premise$_4$, hypothesis)
$x_4 \neq x_7$ cnf(premise$_5$, hypothesis)
$x_4 \neq x_9$ cnf(premise$_6$, hypothesis)
$x_3 \neq x_8$ cnf(premise$_7$, hypothesis)
$x_7 \neq x_{10}$ cnf(premise$_8$, hypothesis)
$x_2 \neq x_6$ cnf(premise$_9$, hypothesis)
$x_2 \neq x_3$ cnf(premise$_{10}$, hypothesis)
$x_2 \neq x_7$ cnf(premise$_{11}$, hypothesis)
heap(sep(lseg(x_5, x_7), sep(lseg(x_2, x_5), sep(lseg(x_2, x_{10}), sep(lseg(x_2, x_1), sep(lseg(x_9, x_1), sep(lseg(x_7, x_6), sep(lseg(x_3, x_{10}), sep
heap(emp) $\Rightarrow x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW426-1.p Randomly generated entailment of the form F → ⊥ (n = 10)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{10}$ cnf(premise$_1$, hypothesis)
$x_6 \neq x_9$ cnf(premise$_2$, hypothesis)
$x_8 \neq x_9$ cnf(premise$_3$, hypothesis)
$x_4 \neq x_{10}$ cnf(premise$_4$, hypothesis)
$x_1 \neq x_6$ cnf(premise$_5$, hypothesis)
$x_1 \neq x_2$ cnf(premise$_6$, hypothesis)
$x_3 \neq x_7$ cnf(premise$_7$, hypothesis)
$x_3 \neq x_{10}$ cnf(premise$_8$, hypothesis)
$x_3 \neq x_5$ cnf(premise$_9$, hypothesis)
$x_9 \neq x_{10}$ cnf(premise$_{10}$, hypothesis)
$x_2 \neq x_7$ cnf(premise$_{11}$, hypothesis)
heap(sep(lseg(x_2, x_4), sep(lseg(x_{10}, x_3), sep(lseg(x_3, x_9), sep(lseg(x_3, x_1), sep(lseg(x_4, x_9), sep(lseg(x_4, x_8), sep(lseg(x_6, x_{10}), em

heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW427-1.p Randomly generated entailment of the form F $\rightarrow \bot$ (n = 11)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_8 \neq x_9$ cnf(premise$_1$, hypothesis)
$x_1 \neq x_{11}$ cnf(premise$_2$, hypothesis)
$x_4 \neq x_{11}$ cnf(premise$_3$, hypothesis)
$x_4 \neq x_7$ cnf(premise$_4$, hypothesis)
$x_3 \neq x_8$ cnf(premise$_5$, hypothesis)
$x_3 \neq x_4$ cnf(premise$_6$, hypothesis)
$x_7 \neq x_8$ cnf(premise$_7$, hypothesis)
$x_7 \neq x_9$ cnf(premise$_8$, hypothesis)
$x_2 \neq x_{11}$ cnf(premise$_9$, hypothesis)
heap(sep(lseg(x_{10}, x_1), sep(lseg(x_9, x_2), sep(lseg(x_9, x_7), sep(lseg(x_7, x_{10}), sep(lseg(x_{11}, x_9), sep(lseg(x_{11}, x_7), sep(lseg(x_{11}, x_3),
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW428-1.p Randomly generated entailment of the form F $\rightarrow \bot$ (n = 11)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_8 \neq x_9$ cnf(premise$_1$, hypothesis)
$x_6 \neq x_8$ cnf(premise$_2$, hypothesis)
$x_6 \neq x_{11}$ cnf(premise$_3$, hypothesis)
$x_6 \neq x_{10}$ cnf(premise$_4$, hypothesis)
$x_4 \neq x_5$ cnf(premise$_5$, hypothesis)
$x_1 \neq x_3$ cnf(premise$_6$, hypothesis)
$x_3 \neq x_4$ cnf(premise$_7$, hypothesis)
$x_9 \neq x_{11}$ cnf(premise$_8$, hypothesis)
$x_2 \neq x_4$ cnf(premise$_9$, hypothesis)
heap(sep(lseg(x_5, x_4), sep(lseg(x_2, x_8), sep(lseg(x_9, x_7), sep(lseg(x_7, x_6), sep(lseg(x_6, x_5), emp)))))) cnf(premise$_{10}$, hypothes
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW429-1.p Randomly generated entailment of the form F $\rightarrow \bot$ (n = 12)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_4 \neq x_7$ cnf(premise$_1$, hypothesis)
$x_4 \neq x_{10}$ cnf(premise$_2$, hypothesis)
$x_3 \neq x_8$ cnf(premise$_3$, hypothesis)
$x_2 \neq x_9$ cnf(premise$_4$, hypothesis)
$x_5 \neq x_9$ cnf(premise$_5$, hypothesis)
heap(sep(lseg(x_5, x_1), sep(lseg(x_7, x_{12}), sep(lseg(x_7, x_8), sep(lseg(x_3, x_5), sep(lseg(x_3, x_{12}), sep(lseg(x_4, x_{12}), sep(lseg(x_4, x_7), se
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW430-1.p Randomly generated entailment of the form F $\rightarrow \bot$ (n = 12)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_3 \neq x_{11}$ cnf(premise$_1$, hypothesis)
$x_3 \neq x_{12}$ cnf(premise$_2$, hypothesis)
$x_7 \neq x_{12}$ cnf(premise$_3$, hypothesis)
$x_2 \neq x_{11}$ cnf(premise$_4$, hypothesis)
$x_2 \neq x_{10}$ cnf(premise$_5$, hypothesis)
heap(sep(lseg(x_5, x_2), sep(lseg(x_5, x_7), sep(lseg(x_5, x_4), sep(lseg(x_{12}, x_1), sep(lseg(x_{12}, x_6), sep(lseg(x_2, x_{12}), sep(lseg(x_9, x_{10}), s
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW431-1.p Randomly generated entailment of the form F → ⊥ (n = 13)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_8 \neq x_{12}$ cnf(premise$_1$, hypothesis)
$x_6 \neq x_7$ cnf(premise$_2$, hypothesis)
$x_6 \neq x_{13}$ cnf(premise$_3$, hypothesis)
$x_3 \neq x_8$ cnf(premise$_4$, hypothesis)
$x_3 \neq x_5$ cnf(premise$_5$, hypothesis)
$x_7 \neq x_8$ cnf(premise$_6$, hypothesis)
$x_7 \neq x_{11}$ cnf(premise$_7$, hypothesis)
$x_{10} \neq x_{11}$ cnf(premise$_8$, hypothesis)
$x_2 \neq x_6$ cnf(premise$_9$, hypothesis)
$x_2 \neq x_3$ cnf(premise$_{10}$, hypothesis)
$x_2 \neq x_9$ cnf(premise$_{11}$, hypothesis)
$x_5 \neq x_7$ cnf(premise$_{12}$, hypothesis)
heap(sep(lseg(x_5, x_3), sep(lseg(x_{10}, x_8), sep(lseg(x_{13}, x_8), sep(lseg(x_1, x_{11}), sep(lseg(x_4, x_9), sep(lseg(x_2, x_9), sep(lseg(x_{12}, x_7), s
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW432-1.p Randomly generated entailment of the form F → ⊥ (n = 13)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_1 \neq x_4$ cnf(premise$_1$, hypothesis)
$x_4 \neq x_6$ cnf(premise$_2$, hypothesis)
$x_7 \neq x_{12}$ cnf(premise$_3$, hypothesis)
$x_2 \neq x_9$ cnf(premise$_4$, hypothesis)
$x_5 \neq x_{13}$ cnf(premise$_5$, hypothesis)
$x_5 \neq x_9$ cnf(premise$_6$, hypothesis)
heap(sep(lseg(x_5, x_9), sep(lseg(x_5, x_1), sep(lseg(x_{13}, x_9), sep(lseg(x_8, x_3), sep(lseg(x_3, x_{11}), sep(lseg(x_{11}, x_{10}), sep(lseg(x_6, x_3), e
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW433-1.p Randomly generated entailment of the form F → ⊥ (n = 14)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_7$ cnf(premise$_1$, hypothesis)
$x_1 \neq x_8$ cnf(premise$_2$, hypothesis)
$x_1 \neq x_{12}$ cnf(premise$_3$, hypothesis)
$x_4 \neq x_{10}$ cnf(premise$_4$, hypothesis)
$x_3 \neq x_4$ cnf(premise$_5$, hypothesis)
$x_7 \neq x_{13}$ cnf(premise$_6$, hypothesis)
$x_{10} \neq x_{11}$ cnf(premise$_7$, hypothesis)
heap(sep(lseg(x_{14}, x_5), sep(lseg(x_{14}, x_7), sep(lseg(x_5, x_2), sep(lseg(x_5, x_{12}), sep(lseg(x_5, x_6), sep(lseg(x_9, x_4), sep(lseg(x_9, x_6), se
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW434-1.p Randomly generated entailment of the form F → ⊥ (n = 14)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{14}$ cnf(premise$_1$, hypothesis)
$x_8 \neq x_{10}$ cnf(premise$_2$, hypothesis)
$x_4 \neq x_8$ cnf(premise$_3$, hypothesis)
$x_4 \neq x_7$ cnf(premise$_4$, hypothesis)
$x_3 \neq x_5$ cnf(premise$_5$, hypothesis)
$x_9 \neq x_{14}$ cnf(premise$_6$, hypothesis)
$x_{13} \neq x_{14}$ cnf(premise$_7$, hypothesis)

$\text{heap}(\text{sep}(\text{lseg}(x_5, x_{10}), \text{sep}(\text{lseg}(x_{13}, x_{12}), \text{sep}(\text{lseg}(x_1, x_7), \text{sep}(\text{lseg}(x_8, x_{14}), \text{sep}(\text{lseg}(x_{12}, x_8), \text{sep}(\text{lseg}(x_2, x_{12}), \text{sep}(\text{lseg}(x_2, x_{11}$
$\text{heap}(\text{emp}) \Rightarrow x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW435-1.p Randomly generated entailment of the form F → ⊥ (n = 15)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{11}$ cnf(premise$_1$, hypothesis)
$x_{11} \neq x_{14}$ cnf(premise$_2$, hypothesis)
$x_3 \neq x_8$ cnf(premise$_3$, hypothesis)
$x_3 \neq x_7$ cnf(premise$_4$, hypothesis)
$x_2 \neq x_8$ cnf(premise$_5$, hypothesis)
$x_1 \neq x_{11}$ cnf(premise$_6$, hypothesis)
$x_4 \neq x_{13}$ cnf(premise$_7$, hypothesis)
$x_{10} \neq x_{11}$ cnf(premise$_8$, hypothesis)
$x_5 \neq x_{12}$ cnf(premise$_9$, hypothesis)
$x_5 \neq x_{15}$ cnf(premise$_{10}$, hypothesis)
$\text{heap}(\text{sep}(\text{lseg}(x_5, x_7), \text{sep}(\text{lseg}(x_2, x_5), \text{sep}(\text{lseg}(x_{12}, x_3), \text{sep}(\text{lseg}(x_9, x_{11}), \text{sep}(\text{lseg}(x_{13}, x_{15}), \text{sep}(\text{lseg}(x_7, x_9), \text{sep}(\text{lseg}(x_7, x_1), \text{s}$
$\text{heap}(\text{emp}) \Rightarrow x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW436-1.p Randomly generated entailment of the form F → ⊥ (n = 15)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{14}$ cnf(premise$_1$, hypothesis)
$x_3 \neq x_6$ cnf(premise$_2$, hypothesis)
$x_3 \neq x_{13}$ cnf(premise$_3$, hypothesis)
$x_4 \neq x_6$ cnf(premise$_4$, hypothesis)
$x_4 \neq x_7$ cnf(premise$_5$, hypothesis)
$x_4 \neq x_{12}$ cnf(premise$_6$, hypothesis)
$x_1 \neq x_3$ cnf(premise$_7$, hypothesis)
$x_{10} \neq x_{15}$ cnf(premise$_8$, hypothesis)
$\text{heap}(\text{sep}(\text{lseg}(x_{13}, x_2), \text{sep}(\text{lseg}(x_4, x_9), \text{sep}(\text{lseg}(x_4, x_{13}), \text{sep}(\text{lseg}(x_1, x_5), \text{sep}(\text{lseg}(x_1, x_6), \text{sep}(\text{lseg}(x_8, x_{14}), \text{sep}(\text{lseg}(x_8, x_{15}), \text{s}$
$\text{heap}(\text{emp}) \Rightarrow x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW437-1.p Randomly generated entailment of the form F → ⊥ (n = 16)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_7$ cnf(premise$_1$, hypothesis)
$x_6 \neq x_9$ cnf(premise$_2$, hypothesis)
$x_{11} \neq x_{16}$ cnf(premise$_3$, hypothesis)
$x_{11} \neq x_{12}$ cnf(premise$_4$, hypothesis)
$x_3 \neq x_6$ cnf(premise$_5$, hypothesis)
$x_3 \neq x_8$ cnf(premise$_6$, hypothesis)
$x_3 \neq x_{11}$ cnf(premise$_7$, hypothesis)
$x_3 \neq x_4$ cnf(premise$_8$, hypothesis)
$x_7 \neq x_8$ cnf(premise$_9$, hypothesis)
$x_7 \neq x_{16}$ cnf(premise$_{10}$, hypothesis)
$x_7 \neq x_{15}$ cnf(premise$_{11}$, hypothesis)
$x_2 \neq x_7$ cnf(premise$_{12}$, hypothesis)
$x_8 \neq x_9$ cnf(premise$_{13}$, hypothesis)
$x_1 \neq x_{11}$ cnf(premise$_{14}$, hypothesis)
$x_1 \neq x_{13}$ cnf(premise$_{15}$, hypothesis)
$x_1 \neq x_{12}$ cnf(premise$_{16}$, hypothesis)
$x_1 \neq x_2$ cnf(premise$_{17}$, hypothesis)
$x_4 \neq x_7$ cnf(premise$_{18}$, hypothesis)
$x_4 \neq x_9$ cnf(premise$_{19}$, hypothesis)

$x_4 \neq x_{14}$ cnf(premise$_{20}$, hypothesis)
$x_{10} \neq x_{16}$ cnf(premise$_{21}$, hypothesis)
$x_{10} \neq x_{12}$ cnf(premise$_{22}$, hypothesis)
$x_{10} \neq x_{14}$ cnf(premise$_{23}$, hypothesis)
$x_{13} \neq x_{16}$ cnf(premise$_{24}$, hypothesis)
$x_{13} \neq x_{14}$ cnf(premise$_{25}$, hypothesis)
$x_5 \neq x_{16}$ cnf(premise$_{26}$, hypothesis)
heap(sep(lseg(x_5, x_1), sep(lseg(x_{12}, x_8), sep(lseg(x_2, x_6), sep(lseg(x_{16}, x_7), sep(lseg(x_{16}, x_3), sep(lseg(x_{10}, x_{15}), sep(lseg(x_{10}, x_5),
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW438-1.p Randomly generated entailment of the form F $\rightarrow \bot$ (n = 16)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{11}$ cnf(premise$_1$, hypothesis)
$x_6 \neq x_7$ cnf(premise$_2$, hypothesis)
$x_3 \neq x_{10}$ cnf(premise$_3$, hypothesis)
$x_3 \neq x_{14}$ cnf(premise$_4$, hypothesis)
$x_7 \neq x_{15}$ cnf(premise$_5$, hypothesis)
$x_9 \neq x_{15}$ cnf(premise$_6$, hypothesis)
$x_9 \neq x_{14}$ cnf(premise$_7$, hypothesis)
$x_2 \neq x_{16}$ cnf(premise$_8$, hypothesis)
$x_2 \neq x_5$ cnf(premise$_9$, hypothesis)
$x_{14} \neq x_{16}$ cnf(premise$_{10}$, hypothesis)
$x_{10} \neq x_{11}$ cnf(premise$_{11}$, hypothesis)
$x_{13} \neq x_{15}$ cnf(premise$_{12}$, hypothesis)
$x_5 \neq x_9$ cnf(premise$_{13}$, hypothesis)
$x_5 \neq x_{16}$ cnf(premise$_{14}$, hypothesis)
$x_5 \neq x_{15}$ cnf(premise$_{15}$, hypothesis)
heap(sep(lseg(x_{14}, x_{12}), sep(lseg(x_2, x_{13}), sep(lseg(x_{13}, x_6), sep(lseg(x_{13}, x_8), sep(lseg(x_{10}, x_2), sep(lseg(x_{10}, x_3), sep(lseg(x_7, x_2
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW439-1.p Randomly generated entailment of the form F $\rightarrow \bot$ (n = 17)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{11}$ cnf(premise$_1$, hypothesis)
$x_6 \neq x_{14}$ cnf(premise$_2$, hypothesis)
$x_3 \neq x_8$ cnf(premise$_3$, hypothesis)
$x_3 \neq x_{15}$ cnf(premise$_4$, hypothesis)
$x_7 \neq x_8$ cnf(premise$_5$, hypothesis)
$x_{14} \neq x_{16}$ cnf(premise$_6$, hypothesis)
$x_8 \neq x_{11}$ cnf(premise$_7$, hypothesis)
$x_8 \neq x_{15}$ cnf(premise$_8$, hypothesis)
$x_1 \neq x_{14}$ cnf(premise$_9$, hypothesis)
$x_{13} \neq x_{15}$ cnf(premise$_{10}$, hypothesis)
$x_{10} \neq x_{11}$ cnf(premise$_{11}$, hypothesis)
$x_{10} \neq x_{13}$ cnf(premise$_{12}$, hypothesis)
$x_5 \neq x_{16}$ cnf(premise$_{13}$, hypothesis)
$x_5 \neq x_9$ cnf(premise$_{14}$, hypothesis)
heap(sep(lseg(x_{13}, x_5), sep(lseg(x_{13}, x_2), sep(lseg(x_{10}, x_3), sep(lseg(x_1, x_{10}), sep(lseg(x_4, x_{12}), sep(lseg(x_2, x_{15}), sep(lseg(x_2, x_{17}
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW440-1.p Randomly generated entailment of the form F $\rightarrow \bot$ (n = 17)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{11}$ cnf(premise$_1$, hypothesis)

$x_6 \neq x_{13}$ cnf(premise$_2$, hypothesis)
$x_{11} \neq x_{17}$ cnf(premise$_3$, hypothesis)
$x_{11} \neq x_{12}$ cnf(premise$_4$, hypothesis)
$x_3 \neq x_{11}$ cnf(premise$_5$, hypothesis)
$x_3 \neq x_{14}$ cnf(premise$_6$, hypothesis)
$x_9 \neq x_{11}$ cnf(premise$_7$, hypothesis)
$x_2 \neq x_{11}$ cnf(premise$_8$, hypothesis)
$x_2 \neq x_7$ cnf(premise$_9$, hypothesis)
$x_2 \neq x_{16}$ cnf(premise$_{10}$, hypothesis)
$x_2 \neq x_{13}$ cnf(premise$_{11}$, hypothesis)
$x_2 \neq x_{12}$ cnf(premise$_{12}$, hypothesis)
$x_{14} \neq x_{16}$ cnf(premise$_{13}$, hypothesis)
$x_1 \neq x_6$ cnf(premise$_{14}$, hypothesis)
$x_1 \neq x_{16}$ cnf(premise$_{15}$, hypothesis)
$x_4 \neq x_6$ cnf(premise$_{16}$, hypothesis)
$x_4 \neq x_{15}$ cnf(premise$_{17}$, hypothesis)
$x_5 \neq x_{14}$ cnf(premise$_{18}$, hypothesis)
heap(sep(lseg(x_{10}, x_5), sep(lseg(x_{10}, x_9), sep(lseg(x_{13}, x_{10}), sep(lseg(x_4, x_{17}), sep(lseg(x_8, x_{12}), sep(lseg(x_{15}, x_{12}), sep(lseg(x_{15}, x
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW441-1.p Randomly generated entailment of the form F $\rightarrow \bot$ (n = 18)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_8$ cnf(premise$_1$, hypothesis)
$x_6 \neq x_9$ cnf(premise$_2$, hypothesis)
$x_6 \neq x_{13}$ cnf(premise$_3$, hypothesis)
$x_6 \neq x_{17}$ cnf(premise$_4$, hypothesis)
$x_6 \neq x_{12}$ cnf(premise$_5$, hypothesis)
$x_3 \neq x_6$ cnf(premise$_6$, hypothesis)
$x_3 \neq x_4$ cnf(premise$_7$, hypothesis)
$x_3 \neq x_{18}$ cnf(premise$_8$, hypothesis)
$x_3 \neq x_{13}$ cnf(premise$_9$, hypothesis)
$x_3 \neq x_{17}$ cnf(premise$_{10}$, hypothesis)
$x_3 \neq x_5$ cnf(premise$_{11}$, hypothesis)
$x_3 \neq x_{15}$ cnf(premise$_{12}$, hypothesis)
$x_7 \neq x_{11}$ cnf(premise$_{13}$, hypothesis)
$x_7 \neq x_{16}$ cnf(premise$_{14}$, hypothesis)
$x_7 \neq x_{15}$ cnf(premise$_{15}$, hypothesis)
$x_9 \neq x_{16}$ cnf(premise$_{16}$, hypothesis)
$x_{17} \neq x_{18}$ cnf(premise$_{17}$, hypothesis)
$x_2 \neq x_8$ cnf(premise$_{18}$, hypothesis)
$x_2 \neq x_{11}$ cnf(premise$_{19}$, hypothesis)
$x_2 \neq x_{18}$ cnf(premise$_{20}$, hypothesis)
$x_2 \neq x_3$ cnf(premise$_{21}$, hypothesis)
$x_2 \neq x_{10}$ cnf(premise$_{22}$, hypothesis)
$x_2 \neq x_{16}$ cnf(premise$_{23}$, hypothesis)
$x_2 \neq x_5$ cnf(premise$_{24}$, hypothesis)
$x_{12} \neq x_{13}$ cnf(premise$_{25}$, hypothesis)
$x_{15} \neq x_{16}$ cnf(premise$_{26}$, hypothesis)
$x_8 \neq x_{11}$ cnf(premise$_{27}$, hypothesis)
$x_8 \neq x_{10}$ cnf(premise$_{28}$, hypothesis)
$x_8 \neq x_{15}$ cnf(premise$_{29}$, hypothesis)
$x_4 \neq x_{18}$ cnf(premise$_{30}$, hypothesis)
$x_4 \neq x_9$ cnf(premise$_{31}$, hypothesis)
$x_4 \neq x_{14}$ cnf(premise$_{32}$, hypothesis)
$x_4 \neq x_{15}$ cnf(premise$_{33}$, hypothesis)
$x_1 \neq x_8$ cnf(premise$_{34}$, hypothesis)
$x_1 \neq x_{11}$ cnf(premise$_{35}$, hypothesis)

$x_1 \neq x_{18}$ cnf(premise$_{36}$, hypothesis)
$x_1 \neq x_{15}$ cnf(premise$_{37}$, hypothesis)
$x_1 \neq x_5$ cnf(premise$_{38}$, hypothesis)
$x_{10} \neq x_{18}$ cnf(premise$_{39}$, hypothesis)
$x_{10} \neq x_{15}$ cnf(premise$_{40}$, hypothesis)
$x_{16} \neq x_{17}$ cnf(premise$_{41}$, hypothesis)
$x_5 \neq x_6$ cnf(premise$_{42}$, hypothesis)
$x_5 \neq x_{16}$ cnf(premise$_{43}$, hypothesis)
heap(sep(lseg(x_5, x_1), sep(lseg(x_{10}, x_{13}), sep(lseg(x_{10}, x_{18}), sep(lseg(x_{18}, x_1), sep(lseg(x_{15}, x_{11}), sep(lseg(x_{14}, x_{17}), sep(lseg(x_{12},
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW442-1.p Randomly generated entailment of the form F $\rightarrow \perp$ (n = 18)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{13}$ cnf(premise$_1$, hypothesis)
$x_6 \neq x_{16}$ cnf(premise$_2$, hypothesis)
$x_{11} \neq x_{18}$ cnf(premise$_3$, hypothesis)
$x_{11} \neq x_{17}$ cnf(premise$_4$, hypothesis)
$x_3 \neq x_{16}$ cnf(premise$_5$, hypothesis)
$x_3 \neq x_{12}$ cnf(premise$_6$, hypothesis)
$x_3 \neq x_{17}$ cnf(premise$_7$, hypothesis)
$x_7 \neq x_{13}$ cnf(premise$_8$, hypothesis)
$x_7 \neq x_{14}$ cnf(premise$_9$, hypothesis)
$x_7 \neq x_{15}$ cnf(premise$_{10}$, hypothesis)
$x_9 \neq x_{13}$ cnf(premise$_{11}$, hypothesis)
$x_9 \neq x_{17}$ cnf(premise$_{12}$, hypothesis)
$x_2 \neq x_8$ cnf(premise$_{13}$, hypothesis)
$x_2 \neq x_6$ cnf(premise$_{14}$, hypothesis)
$x_2 \neq x_{11}$ cnf(premise$_{15}$, hypothesis)
$x_2 \neq x_{17}$ cnf(premise$_{16}$, hypothesis)
$x_{12} \neq x_{14}$ cnf(premise$_{17}$, hypothesis)
$x_8 \neq x_{14}$ cnf(premise$_{18}$, hypothesis)
$x_1 \neq x_{10}$ cnf(premise$_{19}$, hypothesis)
$x_1 \neq x_{15}$ cnf(premise$_{20}$, hypothesis)
$x_4 \neq x_{11}$ cnf(premise$_{21}$, hypothesis)
$x_4 \neq x_9$ cnf(premise$_{22}$, hypothesis)
$x_4 \neq x_{13}$ cnf(premise$_{23}$, hypothesis)
$x_{13} \neq x_{18}$ cnf(premise$_{24}$, hypothesis)
$x_{10} \neq x_{11}$ cnf(premise$_{25}$, hypothesis)
$x_{10} \neq x_{12}$ cnf(premise$_{26}$, hypothesis)
$x_5 \neq x_6$ cnf(premise$_{27}$, hypothesis)
$x_5 \neq x_{16}$ cnf(premise$_{28}$, hypothesis)
heap(sep(lseg(x_5, x_{14}), sep(lseg(x_{13}, x_{15}), sep(lseg(x_{13}, x_{12}), sep(lseg(x_{13}, x_2), sep(lseg(x_{10}, x_{11}), sep(lseg(x_{18}, x_{10}), sep(lseg(x_{18}
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW443-1.p Randomly generated entailment of the form F $\rightarrow \perp$ (n = 19)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{13}$ cnf(premise$_1$, hypothesis)
$x_{11} \neq x_{16}$ cnf(premise$_2$, hypothesis)
$x_{11} \neq x_{13}$ cnf(premise$_3$, hypothesis)
$x_{11} \neq x_{15}$ cnf(premise$_4$, hypothesis)
$x_3 \neq x_4$ cnf(premise$_5$, hypothesis)
$x_3 \neq x_5$ cnf(premise$_6$, hypothesis)
$x_{17} \neq x_{18}$ cnf(premise$_7$, hypothesis)
$x_2 \neq x_8$ cnf(premise$_8$, hypothesis)

$x_2 \neq x_{13}$ cnf(premise$_9$, hypothesis)
$x_2 \neq x_{14}$ cnf(premise$_{10}$, hypothesis)
$x_{14} \neq x_{18}$ cnf(premise$_{11}$, hypothesis)
$x_{14} \neq x_{16}$ cnf(premise$_{12}$, hypothesis)
$x_8 \neq x_9$ cnf(premise$_{13}$, hypothesis)
$x_4 \neq x_{18}$ cnf(premise$_{14}$, hypothesis)
$x_4 \neq x_{19}$ cnf(premise$_{15}$, hypothesis)
$x_1 \neq x_6$ cnf(premise$_{16}$, hypothesis)
$x_1 \neq x_{13}$ cnf(premise$_{17}$, hypothesis)
$x_1 \neq x_{15}$ cnf(premise$_{18}$, hypothesis)
$x_1 \neq x_{14}$ cnf(premise$_{19}$, hypothesis)
$x_{13} \neq x_{19}$ cnf(premise$_{20}$, hypothesis)
$x_5 \neq x_{16}$ cnf(premise$_{21}$, hypothesis)
$x_5 \neq x_{15}$ cnf(premise$_{22}$, hypothesis)
heap(sep(lseg(x_{10}, x_1), sep(lseg(x_{18}, x_{15}), sep(lseg(x_1, x_{14}), sep(lseg(x_1, x_3), sep(lseg(x_{12}, x_{13}), sep(lseg(x_2, x_5), sep(lseg(x_2, x_{12})
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW444-1.p Randomly generated entailment of the form F $\rightarrow \perp$ (n = 19)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{10}$ cnf(premise$_1$, hypothesis)
$x_6 \neq x_{15}$ cnf(premise$_2$, hypothesis)
$x_{11} \neq x_{17}$ cnf(premise$_3$, hypothesis)
$x_3 \neq x_{11}$ cnf(premise$_4$, hypothesis)
$x_3 \neq x_4$ cnf(premise$_5$, hypothesis)
$x_3 \neq x_{18}$ cnf(premise$_6$, hypothesis)
$x_7 \neq x_{16}$ cnf(premise$_7$, hypothesis)
$x_{12} \neq x_{19}$ cnf(premise$_8$, hypothesis)
$x_{17} \neq x_{18}$ cnf(premise$_9$, hypothesis)
$x_2 \neq x_6$ cnf(premise$_{10}$, hypothesis)
$x_2 \neq x_4$ cnf(premise$_{11}$, hypothesis)
$x_2 \neq x_{10}$ cnf(premise$_{12}$, hypothesis)
$x_2 \neq x_{17}$ cnf(premise$_{13}$, hypothesis)
$x_{15} \neq x_{19}$ cnf(premise$_{14}$, hypothesis)
$x_8 \neq x_{11}$ cnf(premise$_{15}$, hypothesis)
$x_8 \neq x_9$ cnf(premise$_{16}$, hypothesis)
$x_8 \neq x_{12}$ cnf(premise$_{17}$, hypothesis)
$x_8 \neq x_{14}$ cnf(premise$_{18}$, hypothesis)
$x_1 \neq x_8$ cnf(premise$_{19}$, hypothesis)
$x_1 \neq x_{13}$ cnf(premise$_{20}$, hypothesis)
$x_{13} \neq x_{17}$ cnf(premise$_{21}$, hypothesis)
$x_{16} \neq x_{19}$ cnf(premise$_{22}$, hypothesis)
$x_5 \neq x_{11}$ cnf(premise$_{23}$, hypothesis)
$x_5 \neq x_{14}$ cnf(premise$_{24}$, hypothesis)
heap(sep(lseg(x_5, x_2), sep(lseg(x_5, x_8), sep(lseg(x_{19}, x_{14}), sep(lseg(x_{19}, x_9), sep(lseg(x_{10}, x_9), sep(lseg(x_{10}, x_{18}), sep(lseg(x_{18}, x_{10}
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW445-1.p Randomly generated entailment of the form F $\rightarrow \perp$ (n = 20)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{19}$ cnf(premise$_1$, hypothesis)
$x_3 \neq x_7$ cnf(premise$_2$, hypothesis)
$x_3 \neq x_{20}$ cnf(premise$_3$, hypothesis)
$x_7 \neq x_{20}$ cnf(premise$_4$, hypothesis)
$x_9 \neq x_{19}$ cnf(premise$_5$, hypothesis)
$x_2 \neq x_{20}$ cnf(premise$_6$, hypothesis)

$x_8 \neq x_{19}$ cnf(premise$_7$, hypothesis)
$x_8 \neq x_{17}$ cnf(premise$_8$, hypothesis)
$x_4 \neq x_{11}$ cnf(premise$_9$, hypothesis)
$x_4 \neq x_{13}$ cnf(premise$_{10}$, hypothesis)
$x_4 \neq x_{19}$ cnf(premise$_{11}$, hypothesis)
$x_1 \neq x_{16}$ cnf(premise$_{12}$, hypothesis)
$x_1 \neq x_{20}$ cnf(premise$_{13}$, hypothesis)
$x_{13} \neq x_{18}$ cnf(premise$_{14}$, hypothesis)
$x_{13} \neq x_{17}$ cnf(premise$_{15}$, hypothesis)
$x_{10} \neq x_{19}$ cnf(premise$_{16}$, hypothesis)
$x_{10} \neq x_{20}$ cnf(premise$_{17}$, hypothesis)
$x_{16} \neq x_{19}$ cnf(premise$_{18}$, hypothesis)
heap(sep(lseg(x_5, x_{17}), sep(lseg(x_{19}, x_1), sep(lseg(x_4, x_{12}), sep(lseg(x_{12}, x_{20}), sep(lseg(x_{12}, x_{15}), sep(lseg(x_{12}, x_{11}), sep(lseg(x_2, x
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW446-1.p Randomly generated entailment of the form F $\rightarrow \perp$ (n = 20)
A randomly generated entailment with n program variables. Negated equalities and list segments are added at random, with specific paramenters so that about half of the generated entailments are valid (or, equivalently, F is unsatisfiable). Normalization and well-formedness axioms should be enough to decide these entailments.
include('Axioms/SWV013-0.ax')
$x_6 \neq x_{13}$ cnf(premise$_1$, hypothesis)
$x_{11} \neq x_{14}$ cnf(premise$_2$, hypothesis)
$x_3 \neq x_7$ cnf(premise$_3$, hypothesis)
$x_7 \neq x_{10}$ cnf(premise$_4$, hypothesis)
$x_7 \neq x_{14}$ cnf(premise$_5$, hypothesis)
$x_9 \neq x_{11}$ cnf(premise$_6$, hypothesis)
$x_{17} \neq x_{19}$ cnf(premise$_7$, hypothesis)
$x_{12} \neq x_{18}$ cnf(premise$_8$, hypothesis)
$x_{12} \neq x_{17}$ cnf(premise$_9$, hypothesis)
$x_2 \neq x_{11}$ cnf(premise$_{10}$, hypothesis)
$x_2 \neq x_{19}$ cnf(premise$_{11}$, hypothesis)
$x_2 \neq x_{17}$ cnf(premise$_{12}$, hypothesis)
$x_{14} \neq x_{16}$ cnf(premise$_{13}$, hypothesis)
$x_1 \neq x_3$ cnf(premise$_{14}$, hypothesis)
$x_4 \neq x_{17}$ cnf(premise$_{15}$, hypothesis)
$x_{10} \neq x_{19}$ cnf(premise$_{16}$, hypothesis)
$x_{10} \neq x_{17}$ cnf(premise$_{17}$, hypothesis)
$x_{10} \neq x_{15}$ cnf(premise$_{18}$, hypothesis)
$x_{13} \neq x_{18}$ cnf(premise$_{19}$, hypothesis)
$x_{13} \neq x_{17}$ cnf(premise$_{20}$, hypothesis)
$x_5 \neq x_7$ cnf(premise$_{21}$, hypothesis)
$x_5 \neq x_{20}$ cnf(premise$_{22}$, hypothesis)
heap(sep(lseg(x_{13}, x_{10}), sep(lseg(x_{16}, x_{10}), sep(lseg(x_{10}, x_{19}), sep(lseg(x_{18}, x_{14}), sep(lseg(x_1, x_{20}), sep(lseg(x_{14}, x_3), sep(lseg(x_{15}
heap(emp) $\Rightarrow$ $x_1 = x_1$ cnf(conclusion$_1$, negated_conjecture)

SWW447-1.p Randomly generated entailment of the form F $\rightarrow$ G (n = 10)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(lseg(x_2, x_7), sep(next(x_6, x_1), sep(next(x_4, x_9), sep(next(x_3, x_2), sep(next(x_8, x_4), sep(lseg(x_{10}, x_2), sep(next(x_5, x_7), s
$\neg$ heap(sep(lseg(x_6, x_1), sep(lseg(x_{10}, x_2), sep(lseg(x_8, x_4), sep(lseg(x_3, x_2), sep(lseg(x_5, x_7), sep(lseg(x_4, x_6), sep(lseg(x_2, x_7), se

SWW448-1.p Randomly generated entailment of the form F $\rightarrow$ G (n = 10)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(next(x_5, x_{10}), sep(lseg(x_9, x_7), sep(next(x_2, x_7), sep(next(x_4, x_7), sep(next(x_6, x_4), sep(next(x_7, x_3), sep(next(x_1, x_2),

$\neg$ heap(sep(lseg(x_3, x_5), sep(lseg(x_8, x_4), sep(lseg(x_9, x_7), sep(lseg(x_6, x_7), sep(lseg(x_5, x_3), emp)))))) cnf(conclusion$_1$, negat

SWW449-1.p Randomly generated entailment of the form F → G (n = 11)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(next(x_1, x_{11}), sep(next(x_3, x_1), sep(next(x_6, x_2), sep(next(x_9, x_6), sep(next(x_4, x_1), sep(next(x_8, x_3), sep(next(x_5, x_{11}
$\neg$ heap(sep(lseg(x_8, x_3), sep(lseg(x_7, x_9), sep(lseg(x_3, x_1), sep(lseg(x_4, x_1), sep(lseg(x_9, x_{11}), sep(lseg(x_{10}, x_{11}), sep(lseg(x_5, x_{11})

SWW450-1.p Randomly generated entailment of the form F → G (n = 11)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(next(x_7, x_1), sep(next(x_{11}, x_7), sep(next(x_9, x_8), sep(lseg(x_4, x_9), sep(next(x_8, x_5), sep(next(x_1, x_4), sep(lseg(x_{10}, x_2),
$\neg$ heap(sep(lseg(x_{11}, x_7), sep(lseg(x_7, x_9), sep(lseg(x_9, x_5), sep(lseg(x_3, x_2), sep(lseg(x_5, x_6), sep(lseg(x_{10}, x_5), emp))))))) cn

SWW451-1.p Randomly generated entailment of the form F → G (n = 12)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(next(x_{10}, x_6), sep(next(x_2, x_1), sep(next(x_8, x_5), sep(next(x_4, x_7), sep(next(x_1, x_6), sep(next(x_9, x_3), sep(next(x_6, x_4)
$\neg$ heap(sep(lseg(x_{10}, x_6), sep(lseg(x_{12}, x_8), sep(lseg(x_{11}, x_5), sep(lseg(x_9, x_5), sep(lseg(x_5, x_1), sep(lseg(x_2, x_4), sep(lseg(x_4, x_6),

SWW452-1.p Randomly generated entailment of the form F → G (n = 12)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(next(x_5, x_6), sep(next(x_{12}, x_4), sep(next(x_8, x_6), sep(lseg(x_1, x_8), sep(next(x_{11}, x_5), sep(lseg(x_4, x_3), sep(next(x_2, x_4),
$\neg$ heap(sep(lseg(x_9, x_6), sep(lseg(x_2, x_4), sep(lseg(x_{12}, x_7), sep(lseg(x_{11}, x_8), sep(lseg(x_1, x_6), sep(lseg(x_7, x_3), emp))))))) cn

SWW453-1.p Randomly generated entailment of the form F → G (n = 13)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(next(x_2, x_6), sep(next(x_4, x_1), sep(next(x_{11}, x_8), sep(lseg(x_1, x_3), sep(next(x_{13}, x_6), sep(next(x_8, x_6), sep(next(x_3, x_7)
$\neg$ heap(sep(lseg(x_4, x_1), sep(lseg(x_9, x_6), sep(lseg(x_5, x_{11}), sep(lseg(x_{12}, x_3), sep(lseg(x_{13}, x_6), sep(lseg(x_{10}, x_{11}), sep(lseg(x_1, x_7

SWW454-1.p Randomly generated entailment of the form F → G (n = 13)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(next(x_1, x_3), sep(next(x_{13}, x_7), sep(next(x_8, x_9), sep(next(x_{12}, x_5), sep(lseg(x_7, x_5), sep(next(x_{10}, x_{13}), sep(lseg(x_9, x_2
$\neg$ heap(sep(lseg(x_{11}, x_3), sep(lseg(x_4, x_{13}), sep(lseg(x_{12}, x_5), sep(lseg(x_1, x_9), sep(lseg(x_6, x_{13}), sep(lseg(x_{10}, x_7), sep(lseg(x_8, x_5

SWW455-1.p Randomly generated entailment of the form F → G (n = 14)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(next(x_3, x_6), sep(next(x_2, x_{12}), sep(lseg(x_1, x_7), sep(next(x_{14}, x_{11}), sep(next(x_8, x_{11}), sep(next(x_{12}, x_1), sep(next(x_{13},

$\neg\,\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_9, x_4), \mathrm{sep}(\mathrm{lseg}(x_{10}, x_5), \mathrm{sep}(\mathrm{lseg}(x_3, x_{11}), \mathrm{sep}(\mathrm{lseg}(x_4, x_{14}), \mathrm{sep}(\mathrm{lseg}(x_5, x_4), \mathrm{emp}))))))$ cnf(conclusion$_1$, ne

SWW456-1.p Randomly generated entailment of the form F → G (n = 14)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
$\mathrm{heap}(\mathrm{sep}(\mathrm{next}(x_{11}, x_8), \mathrm{sep}(\mathrm{next}(x_5, x_{13}), \mathrm{sep}(\mathrm{next}(x_{10}, x_5), \mathrm{sep}(\mathrm{next}(x_9, x_7), \mathrm{sep}(\mathrm{next}(x_7, x_3), \mathrm{sep}(\mathrm{next}(x_3, x_{10}), \mathrm{sep}(\mathrm{next}(x_{12},$
$\neg\,\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_1, x_3), \mathrm{sep}(\mathrm{lseg}(x_2, x_5), \mathrm{sep}(\mathrm{lseg}(x_{14}, x_{12}), \mathrm{sep}(\mathrm{lseg}(x_5, x_{13}), \mathrm{sep}(\mathrm{lseg}(x_6, x_3), \mathrm{sep}(\mathrm{lseg}(x_{11}, x_8), \mathrm{sep}(\mathrm{lseg}(x_8, x_5)$

SWW457-1.p Randomly generated entailment of the form F → G (n = 15)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
$\mathrm{heap}(\mathrm{sep}(\mathrm{next}(x_{13}, x_3), \mathrm{sep}(\mathrm{lseg}(x_{14}, x_2), \mathrm{sep}(\mathrm{lseg}(x_{11}, x_{12}), \mathrm{sep}(\mathrm{next}(x_4, x_{12}), \mathrm{sep}(\mathrm{next}(x_{10}, x_{15}), \mathrm{sep}(\mathrm{next}(x_2, x_1), \mathrm{sep}(\mathrm{next}(x_8$
$\neg\,\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_{14}, x_2), \mathrm{sep}(\mathrm{lseg}(x_{13}, x_3), \mathrm{sep}(\mathrm{lseg}(x_9, x_{11}), \mathrm{sep}(\mathrm{lseg}(x_{10}, x_1), \mathrm{sep}(\mathrm{lseg}(x_5, x_{12}), \mathrm{sep}(\mathrm{lseg}(x_3, x_6), \mathrm{sep}(\mathrm{lseg}(x_1, x_2$

SWW458-1.p Randomly generated entailment of the form F → G (n = 15)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
$\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_{11}, x_{14}), \mathrm{sep}(\mathrm{next}(x_4, x_9), \mathrm{sep}(\mathrm{next}(x_2, x_5), \mathrm{sep}(\mathrm{next}(x_3, x_{12}), \mathrm{sep}(\mathrm{lseg}(x_{14}, x_{11}), \mathrm{sep}(\mathrm{lseg}(x_{10}, x_{13}), \mathrm{sep}(\mathrm{next}(x_6,$
$\neg\,\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_{15}, x_6), \mathrm{sep}(\mathrm{lseg}(x_{10}, x_{13}), \mathrm{sep}(\mathrm{lseg}(x_6, x_2), \mathrm{sep}(\mathrm{lseg}(x_3, x_{12}), \mathrm{sep}(\mathrm{lseg}(x_4, x_9), \mathrm{sep}(\mathrm{lseg}(x_2, x_5), \mathrm{sep}(\mathrm{lseg}(x_9, x_8)$

SWW459-1.p Randomly generated entailment of the form F → G (n = 16)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
$\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_2, x_{12}), \mathrm{sep}(\mathrm{lseg}(x_{10}, x_5), \mathrm{sep}(\mathrm{next}(x_{14}, x_3), \mathrm{sep}(\mathrm{next}(x_1, x_{11}), \mathrm{sep}(\mathrm{next}(x_9, x_7), \mathrm{sep}(\mathrm{next}(x_{16}, x_{10}), \mathrm{sep}(\mathrm{next}(x_8,$
$\neg\,\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_9, x_7), \mathrm{sep}(\mathrm{lseg}(x_6, x_{16}), \mathrm{sep}(\mathrm{lseg}(x_{14}, x_3), \mathrm{sep}(\mathrm{lseg}(x_4, x_3), \mathrm{sep}(\mathrm{lseg}(x_{13}, x_{12}), \mathrm{sep}(\mathrm{lseg}(x_{15}, x_{12}), \mathrm{sep}(\mathrm{lseg}(x_{12},$

SWW460-1.p Randomly generated entailment of the form F → G (n = 16)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
$\mathrm{heap}(\mathrm{sep}(\mathrm{next}(x_3, x_4), \mathrm{sep}(\mathrm{next}(x_8, x_{14}), \mathrm{sep}(\mathrm{next}(x_2, x_{11}), \mathrm{sep}(\mathrm{lseg}(x_5, x_{15}), \mathrm{sep}(\mathrm{next}(x_{14}, x_1), \mathrm{sep}(\mathrm{next}(x_{12}, x_{11}), \mathrm{sep}(\mathrm{next}(x_{1}$
$\neg\,\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_{13}, x_1), \mathrm{sep}(\mathrm{lseg}(x_{12}, x_{11}), \mathrm{sep}(\mathrm{lseg}(x_8, x_{14}), \mathrm{sep}(\mathrm{lseg}(x_{16}, x_{14}), \mathrm{sep}(\mathrm{lseg}(x_3, x_{15}), \mathrm{sep}(\mathrm{lseg}(x_2, x_{11}), \mathrm{sep}(\mathrm{lseg}(x_9,$

SWW461-1.p Randomly generated entailment of the form F → G (n = 17)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
$\mathrm{heap}(\mathrm{sep}(\mathrm{next}(x_7, x_2), \mathrm{sep}(\mathrm{next}(x_6, x_{11}), \mathrm{sep}(\mathrm{next}(x_{12}, x_4), \mathrm{sep}(\mathrm{lseg}(x_9, x_{10}), \mathrm{sep}(\mathrm{next}(x_{10}, x_{13}), \mathrm{sep}(\mathrm{next}(x_8, x_7), \mathrm{sep}(\mathrm{next}(x_{11},$
$\neg\,\mathrm{heap}(\mathrm{sep}(\mathrm{lseg}(x_{16}, x_{14}), \mathrm{sep}(\mathrm{lseg}(x_1, x_{10}), \mathrm{sep}(\mathrm{lseg}(x_{15}, x_2), \mathrm{sep}(\mathrm{lseg}(x_3, x_7), \mathrm{sep}(\mathrm{lseg}(x_{10}, x_{13}), \mathrm{sep}(\mathrm{lseg}(x_{12}, x_4), \mathrm{sep}(\mathrm{lseg}(x_{17},$

SWW462-1.p Randomly generated entailment of the form F → G (n = 17)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
$\mathrm{heap}(\mathrm{sep}(\mathrm{next}(x_{15}, x_2), \mathrm{sep}(\mathrm{lseg}(x_{10}, x_{17}), \mathrm{sep}(\mathrm{next}(x_3, x_5), \mathrm{sep}(\mathrm{lseg}(x_{11}, x_{17}), \mathrm{sep}(\mathrm{next}(x_{12}, x_{13}), \mathrm{sep}(\mathrm{next}(x_6, x_7), \mathrm{sep}(\mathrm{next}(x_{1}$

$\neg$ heap(sep(lseg(x_7, x_{15}), sep(lseg(x_{16}, x_{11}), sep(lseg(x_9, x_1), sep(lseg(x_{12}, x_{13}), sep(lseg(x_{10}, x_{17}), sep(lseg(x_3, x_5), sep(lseg(x_{14},

SWW463-1.p Randomly generated entailment of the form F → G (n = 18)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(lseg(x_{14}, x_{17}), sep(next(x_{17}, x_{13}), sep(next(x_1, x_7), sep(next(x_4, x_6), sep(next(x_{10}, x_8), sep(next(x_8, x_2), sep(lseg(x_3, x
$\neg$ heap(sep(lseg(x_9, x_{12}), sep(lseg(x_{18}, x_{10}), sep(lseg(x_{15}, x_8), sep(lseg(x_{16}, x_7), sep(lseg(x_4, x_6), sep(lseg(x_{12}, x_{14}), sep(lseg(x_3, x

SWW464-1.p Randomly generated entailment of the form F → G (n = 18)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(lseg(x_8, x_6), sep(next(x_6, x_{11}), sep(lseg(x_{15}, x_1), sep(next(x_5, x_6), sep(next(x_2, x_7), sep(lseg(x_{11}, x_4), sep(lseg(x_9, x_5),
$\neg$ heap(sep(lseg(x_8, x_6), sep(lseg(x_{10}, x_4), sep(lseg(x_9, x_4), sep(lseg(x_{15}, x_1), sep(lseg(x_3, x_{18}), sep(lseg(x_2, x_{14}), sep(lseg(x_{12}, x_{16}

SWW465-1.p Randomly generated entailment of the form F → G (n = 19)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(lseg(x_6, x_{15}), sep(lseg(x_4, x_2), sep(next(x_{19}, x_{13}), sep(next(x_8, x_9), sep(next(x_5, x_2), sep(next(x_{15}, x_{13}), sep(next(x_3, x
$\neg$ heap(sep(lseg(x_{10}, x_1), sep(lseg(x_{11}, x_{15}), sep(lseg(x_{14}, x_{18}), sep(lseg(x_{18}, x_4), sep(lseg(x_4, x_2), sep(lseg(x_6, x_{13}), sep(lseg(x_{19},

SWW466-1.p Randomly generated entailment of the form F → G (n = 19)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(lseg(x_6, x_{13}), sep(next(x_8, x_1), sep(next(x_7, x_{16}), sep(next(x_{17}, x_1), sep(lseg(x_{11}, x_{19}), sep(next(x_{16}, x_{11}), sep(next(x_2
$\neg$ heap(sep(lseg(x_8, x_1), sep(lseg(x_9, x_{14}), sep(lseg(x_6, x_2), sep(lseg(x_2, x_5), sep(lseg(x_3, x_{17}), sep(lseg(x_{17}, x_5), sep(lseg(x_{15}, x_{16}

SWW467-1.p Randomly generated entailment of the form F → G (n = 20)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(next(x_6, x_{12}), sep(lseg(x_{15}, x_7), sep(lseg(x_{10}, x_1), sep(next(x_{14}, x_7), sep(next(x_2, x_1), sep(lseg(x_{12}, x_7), sep(next(x_9, x
$\neg$ heap(sep(lseg(x_{15}, x_7), sep(lseg(x_3, x_{12}), sep(lseg(x_8, x_{17}), sep(lseg(x_{14}, x_7), sep(lseg(x_5, x_{17}), sep(lseg(x_7, x_{12}), sep(lseg(x_{10}, x

SWW468-1.p Randomly generated entailment of the form F → G (n = 20)
A randomly generated entailment with n program variables. A random graph with pointers and list segments is generated, and then some of the segments are folded. The task is to prove whether the unfolded version entails the folded one. Parameters are chosen so that about half of the generated entailments are valid. These entailments stress the role of unfolding axioms.
include('Axioms/SWV013-0.ax')
heap(sep(next(x_{14}, x_{13}), sep(next(x_{18}, x_1), sep(next(x_{15}, x_3), sep(next(x_{20}, x_9), sep(next(x_6, x_5), sep(next(x_8, x_{15}), sep(lseg(x_4,
$\neg$ heap(sep(lseg(x_{13}, x_{18}), sep(lseg(x_{20}, x_9), sep(lseg(x_{10}, x_8), sep(lseg(x_2, x_7), sep(lseg(x_6, x_{17}), sep(lseg(x_4, x_3), sep(lseg(x_{12}, x

SWW469+1.p Hoare's Logic with Procedures line 112, 100 axioms selected
is_state(undefined_state(state)) fof(gsy_c_HOL_Oundefined_000tc__Com__Ostate, axiom)
hoare_165779456gleton $\iff$ $\exists s, t$: (is_state(s) and is_state(t) and $s \neq t$) fof(fact_0_state__not__singleton__def, axiom)
$\neg$ induct_false fof(fact_1_induct__false__def, axiom)
induct_true fof(fact_2_induct__trueI, axiom)
induct_true fof(fact_3_induct__true__def, axiom)
hoare_165779456gleton fof(conj$_0$, hypothesis)

$\forall t\colon (\text{is\_state}(t) \Rightarrow \neg\forall s\colon (\text{is\_state}(s) \Rightarrow s = t))$ fof(conj_1, conjecture)

SWW469∧1.p Hoare's Logic with Procedures line 112, 100 axioms selected
state: $tType thf(ty_ty_tc__Com__Ostate, type)
induct_false: $o thf(sy_c_HOL_Oinduct__false, type)
induct_true: $o thf(sy_c_HOL_Oinduct__true, type)
hoare_1821564147gleton: $o thf(sy_c_Hoare__Mirabelle__ghhkfsbqqq_Ostate__not__singleton, type)
hoare_1821564147gleton $\iff \exists s\colon \text{state}, t\colon \text{state}\colon s \neq t$ thf(fact_0_state__not__singleton__def, axiom)
¬ induct_false thf(fact_1_induct__false__def, axiom)
induct_true thf(fact_2_induct__trueI, axiom)
induct_true thf(fact_3_induct__true__def, axiom)
hoare_1821564147gleton thf(conj_0, hypothesis)
$\forall t\colon \text{state}\colon \neg\forall s\colon \text{state}\colon s = t$ thf(conj_1, conjecture)

SWW469_1.p Hoare's Logic with Procedures line 112, 100 axioms selected
state: $tType tff(ty_ty_tc__Com__Ostate, type)
induct_false: $o tff(sy_c_HOL_Oinduct__false, type)
induct_true: $o tff(sy_c_HOL_Oinduct__true, type)
hoare_1310879719gleton: $o tff(sy_c_Hoare__Mirabelle__yiemogtkbg_Ostate__not__singleton, type)
hoare_1310879719gleton $\iff \exists s\colon \text{state}, t\colon \text{state}\colon s \neq t$ tff(fact_0_state__not__singleton__def, axiom)
¬ induct_false tff(fact_1_induct__false__def, axiom)
induct_true tff(fact_2_induct__trueI, axiom)
induct_true tff(fact_3_induct__true__def, axiom)
hoare_1310879719gleton tff(conj_0, hypothesis)
$\forall t\colon \text{state}\colon \neg\forall s\colon \text{state}\colon s = t$ tff(conj_1, conjecture)

SWW581=2.p Checking a large routine-T-WP parameter routine
uni: $tType tff(uni, type)
ty: $tType tff(ty, type)
sort: (ty × uni) → $o tff(sort, type)
witness: ty → uni tff(witness, type)
$\forall a\colon \text{ty}\colon \text{sort}(a, \text{witness}(a))$ tff(witness_sort, axiom)
int: ty tff(int, type)
real: ty tff(real, type)
bool: $tType tff(bool, type)
bool_1: ty tff(bool_1, type)
true: bool tff(true, type)
false: bool tff(false, type)
match_bool: (ty × bool × uni × uni) → uni tff(match_bool, type)
$\forall a\colon \text{ty}, x\colon \text{bool}, x_1\colon \text{uni}, x_2\colon \text{uni}\colon \text{sort}(a, \text{match\_bool}(a, x, x_1, x_2))$ tff(match_bool_sort, axiom)
$\forall a\colon \text{ty}, z\colon \text{uni}, z_1\colon \text{uni}\colon (\text{sort}(a, z) \Rightarrow \text{match\_bool}(a, \text{true}, z, z_1) = z)$ tff(match_bool_True, axiom)
$\forall a\colon \text{ty}, z\colon \text{uni}, z_1\colon \text{uni}\colon (\text{sort}(a, z_1) \Rightarrow \text{match\_bool}(a, \text{false}, z, z_1) = z_1)$ tff(match_bool_False, axiom)
true ≠ false tff(true_False, axiom)
$\forall u\colon \text{bool}\colon (u = \text{true or } u = \text{false})$ tff(bool_inversion, axiom)
tuple_0: $tType tff($\text{tuple}_0$, type)
tuple_{01}: ty tff(tuple_{01}, type)
tuple_{02}: tuple_0 tff(tuple_{02}, type)
$\forall u\colon \text{tuple}_0\colon u = \text{tuple}_{02}$ tff(tuple0_inversion, axiom)
qtmark: ty tff(qtmark, type)
$\forall x\colon \$\text{int}, y\colon \$\text{int}, z\colon \$\text{int}\colon (\$\text{lesseq}(x, y) \Rightarrow (\$\text{lesseq}(0, z) \Rightarrow \$\text{lesseq}(\$\text{product}(x, z), \$\text{product}(y, z))))$ tff(compatOrderMult
fact: $int → $int tff(fact, type)
fact(0) = 1 tff(fact_0, axiom)
$\forall n\colon \$\text{int}\colon (\$\text{lesseq}(1, n) \Rightarrow \text{fact}(n) = \$\text{product}(n, \text{fact}(\$\text{difference}(n, 1))))$ tff(factn, axiom)
ref: ty → ty tff(ref, type)
mk_ref: (ty × uni) → uni tff(mk_ref, type)
$\forall a\colon \text{ty}, x\colon \text{uni}\colon \text{sort}(\text{ref}(a), \text{mk\_ref}(a, x))$ tff(mk_ref_sort, axiom)
contents: (ty × uni) → uni tff(contents, type)
$\forall a\colon \text{ty}, x\colon \text{uni}\colon \text{sort}(a, \text{contents}(a, x))$ tff(contents_sort, axiom)
$\forall a\colon \text{ty}, u\colon \text{uni}\colon (\text{sort}(a, u) \Rightarrow \text{contents}(a, \text{mk\_ref}(a, u)) = u)$ tff(contents_def, axiom)
$\forall a\colon \text{ty}, u\colon \text{uni}\colon (\text{sort}(\text{ref}(a), u) \Rightarrow u = \text{mk\_ref}(a, \text{contents}(a, u)))$ tff(ref_inversion, axiom)

$\forall n$: \$int: (\$lesseq$(0, n)$ $\Rightarrow$ $\forall u$: \$int, r: \$int: ((\$lesseq$(0, r)$ and \$lesseq(r, n) and $u =$ fact(r)) $\Rightarrow$ (\$less(r, n) $\Rightarrow$ $\forall s$: \$int, u_1: \$int: ((\$lesseq$(1, s)$ and \$lesseq$(s,$ \$sum$(r, 1))$ and $u_1 =$ \$product$(s,$ fact$(r)))$ $\Rightarrow$ ($\neg$ \$lesseq(s, r) $\Rightarrow$ $\forall r_1$: \$int: ($r_1 =$ \$sum$(r, 1)$ $\Rightarrow$ (\$lesseq$(0, r_1)$ and \$lesseq(r_1, n) and $u_1 =$ fact(r_1))))))))) tff(wP_parameter_routine, conjec

SWW588=2.p Division-T-WP parameter division
uni: \$tType tff(uni, type)
ty: \$tType tff(ty, type)
sort_1: (ty $\times$ uni) $\rightarrow$ \$o tff(sort, type)
witness_1: ty $\rightarrow$ uni tff(witness, type)
$\forall a$: ty: $\text{sort}_1(a, \text{witness}_1(a))$ tff(witness_sort$_1$, axiom)
int: ty tff(int, type)
real: ty tff(real, type)
bool_1: \$tType tff(bool, type)
bool: ty tff(bool_1, type)
true_1: bool_1 tff(true, type)
false_1: bool_1 tff(false, type)
match_bool_1: (ty $\times$ bool_1 $\times$ uni $\times$ uni) $\rightarrow$ uni tff(match_bool, type)
$\forall a$: ty, x: bool_1, x_1: uni, x_2: uni: $\text{sort}_1(a, \text{match\_bool}_1(a, x, x_1, x_2))$ tff(match_bool_sort$_1$, axiom)
$\forall a$: ty, z: uni, z_1: uni: ($\text{sort}_1(a, z)$ $\Rightarrow$ $\text{match\_bool}_1(a, \text{true}_1, z, z_1) = z$) tff(match_bool_True, axiom)
$\forall a$: ty, z: uni, z_1: uni: ($\text{sort}_1(a, z_1)$ $\Rightarrow$ $\text{match\_bool}_1(a, \text{false}_1, z, z_1) = z_1$) tff(match_bool_False, axiom)
$\text{true}_1 \neq \text{false}_1$ tff(true_False, axiom)
$\forall u$: bool_1: ($u = \text{true}_1$ or $u = \text{false}_1$) tff(bool_inversion, axiom)
tuple_{02}: \$tType tff(tuple_0, type)
tuple_0: ty tff(tuple_{01}, type)
tuple_{03}: tuple_{02} tff(tuple_{02}, type)
$\forall u$: tuple_{02}: $u = \text{tuple}_{03}$ tff(tuple0_inversion, axiom)
qtmark: ty tff(qtmark, type)
$\forall x$: \$int, y: \$int, z: \$int: (\$lesseq(x, y) $\Rightarrow$ (\$lesseq$(0, z)$ $\Rightarrow$ \$lesseq(\$product(x, z), \$product(y, z)))) tff(compatOrderMult
ref: ty $\rightarrow$ ty tff(ref, type)
mk_ref: (ty $\times$ uni) $\rightarrow$ uni tff(mk_ref, type)
$\forall a$: ty, x: uni: $\text{sort}_1(\text{ref}(a), \text{mk\_ref}(a, x))$ tff(mk_ref_sort$_1$, axiom)
contents: (ty $\times$ uni) $\rightarrow$ uni tff(contents, type)
$\forall a$: ty, x: uni: $\text{sort}_1(a, \text{contents}(a, x))$ tff(contents_sort$_1$, axiom)
$\forall a$: ty, u: uni: ($\text{sort}_1(a, u)$ $\Rightarrow$ $\text{contents}(a, \text{mk\_ref}(a, u)) = u$) tff(contents_def$_1$, axiom)
$\forall a$: ty, u: uni: ($\text{sort}_1(\text{ref}(a), u)$ $\Rightarrow$ $u = \text{mk\_ref}(a, \text{contents}(a, u))$) tff(ref_inversion$_1$, axiom)
$\forall a$: \$int, b: \$int: ((\$lesseq$(0, a)$ and \$less$(0, b)$) $\Rightarrow$ (\$sum(\$product$(0, b), a) = a$ and \$lesseq$(0, a)$ and $\forall r$: \$int, q: \$int: ((\$sum(a and \$lesseq$(0, r)$) $\Rightarrow$ ((\$lesseq(b, r) $\Rightarrow$ $\forall q_1$: \$int: ($q_1 =$ \$sum$(q, 1)$ $\Rightarrow$ $\forall r_1$: \$int: ($r_1 =$ \$difference(r, b) $\Rightarrow$ (\$sum(\$product$(q_1, b), r_1) = a$ and \$lesseq$(0, r_1)$ and \$lesseq$(0, r)$ and \$less(r_1, r))))) and ($\neg$ \$lesseq(b, r) $\Rightarrow$ $\exists r_1$: \$int: (\$sum a and \$lesseq$(0, r_1)$ and \$less(r_1, b))))))) tff(wP_parameter_division, conjecture)

SWW673+1.p Priority queue checker
include('Axioms/SWV007+0.ax')
include('Axioms/SWV007+1.ax')
include('Axioms/SWV007+2.ax')
include('Axioms/SWV007+3.ax')
include('Axioms/SWV007+4.ax')

SWW674^1.p ICL logic based upon modal logic based upon simple type theory
include('Axioms/LCL008^0.ax')
include('Axioms/SWV008^0.ax')
include('Axioms/SWV008^1.ax')
include('Axioms/SWV008^2.ax')

SWW675-1.p Lists in Separation Logic
include('Axioms/SWV013-0.ax')