```
;;;; proof of completeness of ground resolution

;;;; Matt Wilding  April 1988

;;;; This proof script executes on Matt Kaufmann's interactive enhancement
;;;; to the Boyer-Moore system

;; This proof script is a proof of the completeness of ground resolution
;; using Bledsoe's excess literal technique.  The final theorem is:

;;      (implies
;;       (and
;;        (unsatisfiable x)
;;        (validclauses x))
;;       (finishedproof x (getproof x)))

;; The proof script is in several sections:

;; 1) definitions required for understanding the theorem
;; 2) definition of getproof (a function that is conjectured to return a valid
;;      resolution proof that ends in box)
;; 3) proof of theorem

;; Section 1 is, strictly speaking, the only section needed to understand what has
;; been proven.
```

EVENT: Start with the initial **nqthm** theory.

```
;; added by Matt Kaufmann just to make nqthm-1991 happy!
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 1: definitions of terms in theorem
```

DEFINITION:
subset $(x,\ y)$
$=$ **if** listp (x)
then if car $(x) \in y$ **then** subset (cdr $(x),\ y)$
else f endif

else t endif

DEFINITION:
set-equal ($s1$, $s2$) $=$ (subset ($s1$, $s2$) $\land$ subset ($s2$, $s1$))

DEFINITION:
set-member (x, $list$)
$=$ **if** listp ($list$)
then if set-equal (x, car ($list$)) **then t**
else set-member (x, cdr ($list$)) **endif**
else f endif

```
;;  define what it means to be a valid set of clauses
;;  (ex.  (validclauses '((a b) ((not a) (not b)) ((not a) b) (a (not b)))) = t)
```

DEFINITION:
validliteral (l)
$=$ ((litatom (l) $\land$ ($l \neq$ **nil**))
$\lor$ (listp (l)
$\land$ (car (l) $=$ `'not`)
$\land$ litatom (cadr (l))
$\land$ (cadr (l) $\neq$ **nil**)
$\land$ (cddr (l) $=$ **nil**)))

DEFINITION:
validclause (c)
$=$ **if** listp (c)
then validclause (cdr (c))
$\land$ validliteral (car (c))
$\land$ (car (c) $\notin$ cdr (c))
else $c =$ **nil endif**

DEFINITION:
validclauses (cs)
$=$ **if** listp (cs) **then** validclause (car (cs)) $\land$ validclauses (cdr (cs))
else $cs =$ **nil endif**

```
;; unsatisfiablility of a set of clauses
```

DEFINITION:
unsatwith ($clause$, cs, $values$)
$=$ **if** listp ($clause$)
then if (list (`'not`, car ($clause$)) $\in$ $values$)
$\lor$ (cadar ($clause$) $\in$ $values$)

then unsatwith (cdr (*clause*), *cs*, *values*)
else unsatwith (cdr (*clause*), *cs*, *values*)
$\wedge$ **if** listp (*cs*)
then unsatwith (car (*cs*),
cdr (*cs*),
cons (car (*clause*), *values*))
else f endif endif
else t endif

Definition:
unsatisfiable (*cs*)
= **if** listp (*cs*) **then** unsatwith (car (*cs*), cdr (*cs*), **nil**)
else f endif

Definition:
takeout (*list*, *x*)
= **if** listp (*list*)
then if x = car (*list*) **then** takeout (cdr (*list*), *x*)
else cons (car (*list*), takeout (cdr (*list*), *x*)) **endif**
else nil endif

```
;; resolvent is valid resolvent of p1 and p2 resolving on something in p1list
```

Definition:
resolvent-help (*resolvent*, *p1list*, *p1*, *p2*)
= **if** listp (*p1list*)
then if list (`'not`, car (*p1list*)) $\in$ *p2*
then subset (takeout (*p1*, car (*p1list*))
$\cup$ takeout (*p2*, list (`'not`, car (*p1list*))),
resolvent)
$\vee$ resolvent-help (*resolvent*, cdr (*p1list*), *p1*, *p2*)
elseif cadar (*p1list*) $\in$ *p2*
then subset (takeout (*p1*, car (*p1list*))
$\cup$ takeout (*p2*, cadar (*p1list*)),
resolvent)
$\vee$ resolvent-help (*resolvent*, cdr (*p1list*), *p1*, *p2*)
else resolvent-help (*resolvent*, cdr (*p1list*), *p1*, *p2*) **endif**
else f endif

Definition: resolvent (*r*, *p1*, *p2*) = resolvent-help (*r*, *p1*, *p1*, *p2*)

```
;; a valid resolution proof for a set of axioms is a set of triples where
;;each line of the proof has a resolvent and two (axiomatic or derived) parents
```

Definition:

validproof$(axioms, proof)$
$=$ **if** listp$(proof)$
then set-member (cadar $(proof)$, $axioms$)
$\wedge$ set-member (caddar $(proof)$, $axioms$)
$\wedge$ resolvent (caar $(proof)$, cadar $(proof)$, caddar $(proof)$)
$\wedge$ validproof (cons (caar $(proof)$, $axioms$), cdr $(proof)$)
else t endif

```
;; the empty clause is in a set of clauses
```

DEFINITION:
box-in-axioms $(axioms)$
$=$ **if** listp $(axioms)$
then if listp (car $(axioms)$) **then** box-in-axioms (cdr $(axioms)$)
else t endif
else f endif

DEFINITION:
last-element $(list)$
$=$ **if** listp $(list)$
then if listp (cdr $(list)$) **then** last-element (cdr $(list)$)
else car $(list)$ **endif**
else nil endif

```
;; proof is a finished proof of axioms if there are no axioms, if
;; there are no axioms (and therefore satisfiable), if axioms include box,
;; or if proof is valid and has as its last resolvent box.
```

DEFINITION:
finishedproof $(axioms, proof)$
$=$ (($\neg$ listp $(axioms)$)
$\vee$ box-in-axioms $(axioms)$
$\vee$ (validproof $(axioms, proof)$ $\wedge$ (car (last-element $(proof)$) $\simeq$ **nil**)))

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 2:  getproof function developement
```

DEFINITION:
length $(list)$
$=$ **if** listp $(list)$ **then** $1 +$ length (cdr $(list)$)
else 0 **endif**

THEOREM: list-means-length-more-0
listp $(x) \rightarrow (0 <$ length $(x))$

DEFINITION:
takeout1 ($list$, x)
= **if** listp ($list$)
then if x = car ($list$) **then** cdr ($list$)
else cons (car ($list$), takeout1 (cdr ($list$), x)) **endif**
else nil endif

DEFINITION:
listofcars ($list$)
= **if** listp ($list$) **then** cons (caar ($list$), listofcars (cdr ($list$)))
else nil endif

```
;; some really useful rewrite rules for sets
```

THEOREM: membership-on-subsets
$((e \in a) \land \text{subset}(a, b)) \rightarrow (e \in b)$

THEOREM: equals-has-same-members
$\text{set-equal}(a, b) \rightarrow ((e \in a) = (e \in b))$

THEOREM: subset-is-transitive
$(\text{subset}(x, y) \land \text{subset}(y, z)) \rightarrow \text{subset}(x, z)$

THEOREM: subset-fact-1
$\text{subset}(x, y) \rightarrow \text{subset}(x, \text{cons}(a, y))$

THEOREM: set-equals-fact
$\text{set-equal}(x, y) \rightarrow \text{set-equal}(\text{cons}(a, x), \text{cons}(a, y))$

THEOREM: subset-of-self
$\text{subset}(x, x)$

THEOREM: set-equal-self
$\text{set-equal}(x, x)$

THEOREM: member-is-setmember
$(a \in x) \rightarrow \text{set-member}(a, x)$

THEOREM: setmember-is-setmember-of-superset
$(\text{set-member}(a, x) \land \text{subset}(x, y)) \rightarrow \text{set-member}(a, y)$

THEOREM: takeout-fact1
$\text{subset}(\text{takeout}(s, x), \text{takeout}(\text{cons}(a, s), x))$

THEOREM: subset-of-union-fact-1
$\text{subset}(a \cup b, c) = (\text{subset}(a, c) \land \text{subset}(b, c))$

THEOREM: takeout-fact2
$(a \notin x) \rightarrow$ set-equal (takeout $(x,\ a),\ x)$

THEOREM: stupid-lemma1
$(l \in c) \rightarrow$ (subset (cons $(l,\ c),\ d)$ = subset $(c,\ d))$

THEOREM: stupid-lemma2
$(l \in d) \rightarrow$ (subset $(c,$ cons $(l,\ d))$ = subset $(c,\ d))$

THEOREM: stupid-lemma3
$(l \in c) \rightarrow$ (set-member (cons $(l,\ c),\ d)$ = set-member $(c,\ d))$

```
;; some useful oddball facts about append other things
```

THEOREM: member-of-append-1
$(a \in x) \rightarrow (a \in$ append $(x,\ y))$

THEOREM: is-last-element-means-member
(listp $(p) \wedge$ (car (last-element $(p)) = x)) \rightarrow (x \in$ listofcars $(p))$

THEOREM: membership-of-append
$(a \in x) \rightarrow (a \in$ append $(y,\ x))$

THEOREM: subset-of-append
subset $(x,\ y) \rightarrow$ subset $(x,$ append $(z,\ y))$

DEFINITION:
first-non-unit (*clauses*)
= **if** listp (*clauses*)
then if 1 < length (car (*clauses*)) **then** car (*clauses*)
else first-non-unit (cdr (*clauses*)) **endif**
else nil endif

THEOREM: first-non-unit-fact2
(length (first-non-unit $(x)) = 1) =$ **f**

DEFINITION:
take-out-literal $(cs,\ c)$
= **if** listp (cs)
then if $c =$ car (cs) **then** cons (cdr (c), cdr (cs))
else cons (car (cs), take-out-literal (cdr $(cs),\ c)$) **endif**
else nil endif

DEFINITION:
take-out-clause $(cs,\ c)$
= **if** listp (cs)
then if $c =$ car (cs) **then** cons (list (car (c)), cdr (cs))
else cons (car (cs), take-out-clause (cdr $(cs),\ c)$) **endif**
else nil endif

THEOREM: subset-of-takeouts-fact
(validclauses (x)
 $\land$ $(a \in x)$
 $\land$ (list (car (a)) $\in y$)
 $\land$ subset (takeout1 $(x, a), y$))
$\rightarrow$ subset (take-out-clause $(x, a), y$)

THEOREM: take-out-literal-communative-sort-of
$(a \in x)$
$\rightarrow$ set-equal (take-out-literal (cons $(y, x), a$), cons (y, take-out-literal (x, a)))

THEOREM: take-out-literal-preserves-validclauseness
(($\neg$ box-in-axioms (cs)) $\land$ validclauses (cs))
$\rightarrow$ validclauses (take-out-literal (cs, x))

THEOREM: take-out-clause-preserves-validclauseness
(($\neg$ box-in-axioms (cs)) $\land$ validclauses (cs))
$\rightarrow$ validclauses (take-out-clause (cs, x))

DEFINITION:
find-contradiction (cs)
= **if** listp (cs)
  then if length (car (cs)) = `1`
    then if list (list (`'not`, caar (cs))) $\in$ cdr (cs)
      then list (**nil**, car (cs), list (list (`'not`, caar (cs))))
      elseif list (cadaar (cs)) $\in$ cdr (cs)
      then list (**nil**, car (cs), list (cadaar (cs)))
      else find-contradiction (cdr (cs)) **endif**
    else list (**nil**, **nil**, **nil**) **endif**
  else nil endif

DEFINITION:
number-of-lits (cs)
= **if** listp (cs) **then** length (car (cs)) + number-of-lits (cdr (cs))
  else `0` **endif**

DEFINITION:
excess-literal (cs) = (number-of-lits (cs) $-$ length (cs))

THEOREM: first-non-unit-fact
((excess-literal $(cs) \not\simeq$ `0`) $\land$ ($\neg$ box-in-axioms (cs)))
$\rightarrow$ ((`1` < length (first-non-unit (cs))) = **t**)

DEFINITION:
adjust (*proof*, cs, l)

$$
\begin{array}{ll}
= & \textbf{if } \mathrm{listp}\,(\mathit{proof}) \\
& \textbf{then if } \text{set-member}\,(\mathrm{cadar}\,(\mathit{proof}),\ \mathit{cs}) \\
& \quad \textbf{then if } \text{set-member}\,(\mathrm{caddar}\,(\mathit{proof}),\ \mathit{cs}) \\
& \quad\quad \textbf{then } \mathrm{cons}\,(\mathrm{car}\,(\mathit{proof}), \\
& \quad\quad\quad\quad \mathrm{adjust}\,(\mathrm{cdr}\,(\mathit{proof}),\ \mathrm{cons}\,(\mathrm{caar}\,(\mathit{proof}),\ \mathit{cs}),\ l)) \\
& \quad\quad \textbf{else } \mathrm{cons}\,(\mathrm{list}\,(\text{add-to-set}\,(l,\ \mathrm{caar}\,(\mathit{proof})), \\
& \quad\quad\quad\quad\quad \mathrm{cadar}\,(\mathit{proof}), \\
& \quad\quad\quad\quad\quad \text{add-to-set}\,(l,\ \mathrm{caddar}\,(\mathit{proof}))), \\
& \quad\quad\quad\quad \mathrm{adjust}\,(\mathrm{cdr}\,(\mathit{proof}), \\
& \quad\quad\quad\quad\quad \mathrm{cons}\,(\text{add-to-set}\,(l,\ \mathrm{caar}\,(\mathit{proof})),\ \mathit{cs}), \\
& \quad\quad\quad\quad\quad l))\ \textbf{endif} \\
& \quad \textbf{elseif } \text{set-member}\,(\mathrm{caddar}\,(\mathit{proof}),\ \mathit{cs}) \\
& \quad \textbf{then } \mathrm{cons}\,(\mathrm{list}\,(\text{add-to-set}\,(l,\ \mathrm{caar}\,(\mathit{proof})), \\
& \quad\quad\quad\quad \text{add-to-set}\,(l,\ \mathrm{cadar}\,(\mathit{proof})), \\
& \quad\quad\quad\quad \mathrm{caddar}\,(\mathit{proof})), \\
& \quad\quad\quad \mathrm{adjust}\,(\mathrm{cdr}\,(\mathit{proof}),\ \mathrm{cons}\,(\text{add-to-set}\,(l,\ \mathrm{caar}\,(\mathit{proof})),\ \mathit{cs}),\ l)) \\
& \quad \textbf{else } \mathrm{cons}\,(\mathrm{list}\,(\text{add-to-set}\,(l,\ \mathrm{caar}\,(\mathit{proof})), \\
& \quad\quad\quad\quad \text{add-to-set}\,(l,\ \mathrm{cadar}\,(\mathit{proof})), \\
& \quad\quad\quad\quad \text{add-to-set}\,(l,\ \mathrm{caddar}\,(\mathit{proof}))), \\
& \quad\quad\quad \mathrm{adjust}\,(\mathrm{cdr}\,(\mathit{proof}), \\
& \quad\quad\quad\quad \mathrm{cons}\,(\text{add-to-set}\,(l,\ \mathrm{caar}\,(\mathit{proof})),\ \mathit{cs}), \\
& \quad\quad\quad\quad l))\ \textbf{endif} \\
& \textbf{else nil endif}
\end{array}
$$

THEOREM: adjust-is-listp-when

$$\mathrm{listp}\,(p) \rightarrow \mathrm{listp}\,(\mathrm{adjust}\,(p,\ x,\ l))$$

THEOREM: when-first-non-unit-gives-clause

$$
\begin{array}{l}
((\neg\ \text{box-in-axioms}\,(\mathit{cs})) \wedge (\text{excess-literal}\,(\mathit{cs}) \not\simeq \mathsf{0})) \\
\rightarrow \quad (\text{first-non-unit}\,(\mathit{cs}) \in \mathit{cs})
\end{array}
$$

THEOREM: no-box-fact

$$(\neg\ \text{box-in-axioms}\,(\mathit{cs})) \rightarrow (\text{number-of-lits}\,(\mathit{cs}) \not< \mathrm{length}\,(\mathit{cs}))$$

THEOREM: excess-literal-in-terms-of-cdr

$$
\begin{array}{l}
(\mathrm{listp}\,(\mathit{cs}) \wedge (\neg\ \text{box-in-axioms}\,(\mathit{cs}))) \\
\rightarrow \quad (\text{excess-literal}\,(\mathit{cs}) \\
\qquad\quad = \quad ((\mathrm{length}\,(\mathrm{car}\,(\mathit{cs})) + \text{excess-literal}\,(\mathrm{cdr}\,(\mathit{cs}))) - 1))
\end{array}
$$

THEOREM: when-take-out-literal-does-something

$$
\begin{array}{l}
((\neg\ \text{box-in-axioms}\,(\mathit{cs})) \wedge (\text{excess-literal}\,(\mathit{cs}) \not\simeq \mathsf{0})) \\
\rightarrow \quad ((\text{excess-literal}\,(\text{take-out-literal}\,(\mathit{cs},\ \text{first-non-unit}\,(\mathit{cs}))) \\
\qquad\quad < \quad \text{excess-literal}\,(\mathit{cs})) \\
\qquad\quad = \quad \mathbf{t})
\end{array}
$$

THEOREM: not-box-when-clause-taken
$((\neg\ \text{box-in-axioms}\,(cs)) \land (\mathtt{1} < \text{length}\,(x)))$
$\rightarrow\quad (\neg\ \text{box-in-axioms}\,(\text{take-out-clause}\,(cs,\ x)))$

THEOREM: no-box-zero-el-fact
$(\text{listp}\,(x) \land (\neg\ \text{box-in-axioms}\,(x)) \land (\text{excess-literal}\,(x) \simeq \mathtt{0}))$
$\rightarrow\quad (\text{excess-literal}\,(\text{cdr}\,(x)) \simeq \mathtt{0})$

THEOREM: take-out-clause-fact
$((\neg\ \text{box-in-axioms}\,(cs)) \land (\mathtt{1} < \text{length}\,(x)) \land (x \in cs))$
$\rightarrow\quad ((\text{excess-literal}\,(\text{take-out-clause}\,(cs,\ x)) < \text{excess-literal}\,(cs)) = \mathbf{t})$

THEOREM: when-take-out-clause-does-something
$((\neg\ \text{box-in-axioms}\,(cs)) \land (\text{excess-literal}\,(cs) \not\simeq \mathtt{0}))$
$\rightarrow\quad ((\text{excess-literal}\,(\text{take-out-clause}\,(cs,\ \text{first-non-unit}\,(cs)))$
$\quad\quad < \quad \text{excess-literal}\,(cs))$
$\quad = \quad \mathbf{t})$

DEFINITION:
getproof (cs)
$=$ **if** box-in-axioms (cs) **then nil**
elseif excess-literal $(cs) \simeq \mathtt{0}$ **then** list (find-contradiction (cs))
elseif car (last-element (adjust (getproof (take-out-literal (cs,
first-non-unit (cs))),
cs,
car (first-non-unit (cs)))))
$=$ **nil**
then adjust (getproof (take-out-literal (cs, first-non-unit (cs))),
cs,
car (first-non-unit (cs)))
else append (adjust (getproof (take-out-literal (cs,
first-non-unit (cs))),
cs,
car (first-non-unit (cs))),
getproof (take-out-clause (cs, first-non-unit (cs)))) **endif**

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Section 3. Proof of the completeness theorem


;; the format of a resolution proof (implicit in validproof but useful to make explicit
```

DEFINITION:
proof-form (p)
$=$ **if** listp (p)

then validclause (caar (p))
$\quad\wedge$ validclause (cadar (p))
$\quad\wedge$ validclause (caddar (p))
$\quad\wedge$ (cdddar (p) = **nil**)
$\quad\wedge$ proof-form (cdr (p))
else p = **nil endif**

```
;; some useful facts about valid proofs
```

THEOREM: validproof-on-subset
(subset (x, y) $\wedge$ validproof (x, p)) $\rightarrow$ validproof (y, p)

THEOREM: validproof-clauses-commutative
set-equal (x, y) $\rightarrow$ (validproof (x, p) = validproof (y, p))

```
;; A useful fact about unsatisfiable sets of clauses
```

THEOREM: assignment-interchangeable
(unsatwith (c, cs, z) $\wedge$ set-equal (y, z)) $\rightarrow$ unsatwith (c, cs, y)

```
;; lemmas about preservation of unsatisfiablility of clauses with some of
;; the literals removed
```

THEOREM: take-out-literal-helper
unsatwith ($clause$, cs, v)
$\rightarrow$ unsatwith (car (take-out-literal (cons ($clause$, cs), x)),
cdr (take-out-literal (cons ($clause$, cs), x)),
v)

THEOREM: without-literal-still-unsatisfiable
(unsatisfiable (cs) $\wedge$ listp (cs)) $\rightarrow$ unsatisfiable (take-out-literal (cs, x))

THEOREM: unsat-stepper
(listp (x) $\wedge$ unsatwith (x, cs, v)) $\rightarrow$ unsatwith (list (car (x)), cs, v)

THEOREM: take-out-clause-helper
(listp (x) $\wedge$ unsatwith ($clause$, cs, v))
$\rightarrow$ unsatwith (car (take-out-clause (cons ($clause$, cs), x)),
cdr (take-out-clause (cons ($clause$, cs), x)),
v)

THEOREM: when-unsatisfiable
unsatwith ($clause$, cs, z) $\rightarrow$ unsatwith ($clause$, cs, cons (x, z))

DEFINITION:
unsat-induct $(c,\ x,\ cs,\ v)$
$=$ **if** listp (cs)
then if $(x = \text{cadr}(\text{car}(c))) \lor (x = \text{list}(\texttt{'not}, \text{car}(c)))$ **then t**
else unsat-induct $(\text{car}(cs),\ x,\ \text{cdr}(cs),\ \text{cons}(\text{car}(c),\ v))$ **endif**
else t endif

```
;; useful facts about negations of literals
```

THEOREM: reversible-nots-1
$((\text{length}(y) = \texttt{1})$
$\land$ litatom (x)
$\land$ $(x \neq \textbf{nil})$
$\land$ validclause (y)
$\land$ $(x = \text{cadar}(y)))$
$\rightarrow$ $(\text{list}(\text{list}(\texttt{'not}, x)) = y)$

THEOREM: reversible-nots-2
$((\text{length}(y) = \texttt{1})$
$\land$ listp (x)
$\land$ validclause (y)
$\land$ $(x = \text{list}(\texttt{'not}, \text{car}(y))))$
$\rightarrow$ $(\text{list}(\text{cadr}(x)) = y)$

```
;; useful definition of set of clauses with all unit clauses
```

DEFINITION:
allunits (x)
$=$ **if** listp (x) **then** $(\text{length}(\text{car}(x)) = \texttt{1}) \land \text{allunits}(\text{cdr}(x))$
else t endif

THEOREM: allunit-means
$\text{allunits}(x) \rightarrow (\neg\ \text{listp}(\text{cdar}(x)))$

THEOREM: allunits-if
$((\neg\ \text{box-in-axioms}(x)) \land (\text{excess-literal}(x) \simeq \texttt{0})) \rightarrow \text{allunits}(x)$

THEOREM: unit-clause-fact
$(\text{length}(x) = \texttt{1}) \rightarrow (\text{takeout}(x,\ \text{car}(x)) = \textbf{nil})$

THEOREM: restriction-list-commutative
$\text{set-equal}(z1,\ z2) \rightarrow (\text{unsatwith}(x,\ y,\ z1) = \text{unsatwith}(x,\ y,\ z2))$

THEOREM: unsat-ignore-first-helper
$((\text{litatom}(x) \rightarrow (\text{list}(\text{list}(\texttt{'not}, x)) \notin \text{cons}(c,\ cs)))$
$\land$ $(\text{listp}(x) \rightarrow (\text{list}(\text{cadr}(x)) \notin \text{cons}(c,\ cs)))$

$\land$ allunits (cons (c, cs))
$\land$ validclause (list (x))
$\land$ validclauses (cons (c, cs))
$\land$ unsatwith (c, cs, cons (x, v)))
$\rightarrow$ unsatwith (c, cs, v)

```
;; base case of main theorem - if all clauses unit then getproof finds a proof
```

THEOREM: no-excess-literal-proof
(validclauses (cs)
$\land$ ($\neg$ box-in-axioms (cs))
$\land$ (excess-literal (cs) $\simeq$ 0)
$\land$ unsatisfiable (cs))
$\rightarrow$ finishedproof (cs, list (find-contradiction (cs)))

```
;; useful definition of "almost" valid proof.  p is validproof of cs except that
;; some clauses in proof are missing literal l
```

DEFINITION:
validproof-but-for-literal (cs, p, l)
= **if** listp (p)
then (set-member (cadar (p), cs)
$\lor$ set-member (add-to-set (l, cadar (p)), cs))
$\land$ (set-member (caddar (p), cs)
$\lor$ set-member (add-to-set (l, caddar (p)), cs))
$\land$ resolvent (caar (p), cadar (p), caddar (p))
$\land$ validproof-but-for-literal (cons (caar (p), cs), cdr (p), l)
else t endif

```
;; some facts about validproof-but-for-literal
```

THEOREM: validproof-but-for-literal-clauses-commutative
set-equal (x, y)
$\rightarrow$ (validproof-but-for-literal (x, p, l)
= validproof-but-for-literal (y, p, l))

DEFINITION:
adding-literal-induct (cs, p)
= **if** listp (p) **then** adding-literal-induct (cons (caar (p), cs), cdr (p))
else t endif

THEOREM: adding-literal-fact
validproof-but-for-literal (cons (x, cs), p, l)
$\rightarrow$ validproof-but-for-literal (cons (add-to-set (l, x), cs), p, l)

```
;; facts about adjusting clauses by adding literals and still getting valid resolvents
```

THEOREM: resolvent-with-extra-literal-fact1b
(validliteral (l)
$\wedge$ $(l \notin \mathit{p2})$
$\wedge$ $(l \in r)$
$\wedge$ validclause $(\mathit{p1})$
$\wedge$ validclause $(\mathit{p2})$
$\wedge$ validclause $(\mathit{p1list})$
$\wedge$ resolvent-help $(r,\ \mathit{p1list},\ \mathit{p1},\ \mathit{p2}))$
$\rightarrow$ resolvent-help $(r,\ \mathit{p1list},\ \mathit{p1},\ \text{cons}\,(l,\ \mathit{p2}))$

THEOREM: resolvent-with-extra-literal-fact1c
(validliteral (l)
$\wedge$ $(l \notin r)$
$\wedge$ $(l \notin \mathit{p2})$
$\wedge$ validclause $(\mathit{p1})$
$\wedge$ validclause $(\mathit{p2})$
$\wedge$ validclause $(\mathit{p1list})$
$\wedge$ resolvent-help $(r,\ \mathit{p1list},\ \mathit{p1},\ \mathit{p2}))$
$\rightarrow$ resolvent-help $(\text{cons}\,(l,\ r),\ \mathit{p1list},\ \mathit{p1},\ \text{cons}\,(l,\ \mathit{p2}))$

THEOREM: resolvent-with-extra-literal-fact2a
$((l \notin r)$
$\wedge$ $(l \notin \mathit{p1})$
$\wedge$ validliteral (l)
$\wedge$ validclause $(\mathit{p1})$
$\wedge$ validclause $(\mathit{p2})$
$\wedge$ validclause $(\mathit{p1list})$
$\wedge$ resolvent-help $(r,\ \mathit{p1list},\ \mathit{p1},\ \mathit{p2}))$
$\rightarrow$ resolvent-help $(\text{cons}\,(l,\ r),\ \mathit{p1list},\ \text{cons}\,(l,\ \mathit{p1}),\ \mathit{p2})$

THEOREM: resolvent-with-extra-literal-fact2b
$((l \in r)$
$\wedge$ $(l \notin \mathit{p1})$
$\wedge$ validliteral (l)
$\wedge$ validclause $(\mathit{p1})$
$\wedge$ validclause $(\mathit{p2})$
$\wedge$ validclause $(\mathit{p1list})$
$\wedge$ resolvent-help $(r,\ \mathit{p1list},\ \mathit{p1},\ \mathit{p2}))$
$\rightarrow$ resolvent-help $(r,\ \mathit{p1list},\ \text{cons}\,(l,\ \mathit{p1}),\ \mathit{p2})$

THEOREM: resolvent-with-extra-literal-fact3a
$((l \notin r)$
$\wedge$ $(l \notin \mathit{p1})$
$\wedge$ $(l \notin \mathit{p2})$
$\wedge$ validliteral (l)

$\wedge$ validclause $(p1)$
$\wedge$ validclause $(p2)$
$\wedge$ validclause $(p1list)$
$\wedge$ resolvent-help $(r,\ p1list,\ p1,\ p2))$
$\rightarrow$ resolvent-help $(\text{cons}(l,\ r),\ p1list,\ \text{cons}(l,\ p1),\ \text{cons}(l,\ p2))$

THEOREM: resolvent-fact
resolvent-help $(a,\ b,\ c,\ d) \rightarrow$ resolvent-help $(\text{cons}(l,\ a),\ b,\ c,\ d)$

THEOREM: extra-try-resolvent-fact
resolvent-help $(a,\ b,\ c,\ d) \rightarrow$ resolvent-help $(a,\ \text{cons}(l,\ b),\ c,\ d)$

THEOREM: when-not-member-of-take-out-literal
$((1 < \text{length}(c)) \wedge (a \in \text{take-out-literal}(x,\ c)) \wedge (a \notin x))$
$\rightarrow$ $(\text{cons}(\text{car}(c),\ a) \in x)$

THEOREM: take-out-literal-is-validclause
$(\text{validclauses}(x) \wedge (1 < \text{length}(c)))$
$\rightarrow$ validclauses $(\text{take-out-literal}(x,\ c))$

THEOREM: member-of-validclause-is-validlit
$(\text{validclause}(cs) \wedge (l \in cs)) \rightarrow \text{validliteral}(l)$

THEOREM: when-literal-not-there
$(\text{validclauses}(cs) \wedge (x \notin cs)) \rightarrow (\text{take-out-literal}(cs,\ x) = cs)$

THEOREM: real-proof-implies-almost-proof
validproof $(cs,\ p) \rightarrow$ validproof-but-for-literal $(cs,\ p,\ x)$

THEOREM: when-not-in-bigger-clauses
$((\neg\ \text{set-member}(x,\ cs)) \wedge \text{set-member}(x,\ \text{take-out-literal}(cs,\ p)))$
$\rightarrow$ set-equal $(\text{cdr}(p),\ x)$

THEOREM: take-out-literal-produces-almost-proof
(proof-form (p)
$\wedge$ validproof $(\text{take-out-literal}(cs,\ c),\ p)$
$\wedge$ validclauses (cs)
$\wedge$ $(1 < \text{length}(c)))$
$\rightarrow$ validproof-but-for-literal $(cs,\ p,\ \text{car}(c))$

THEOREM: member-of-validclauses-is-validclause
$(\text{validclauses}(x) \wedge (a \in x)) \rightarrow \text{validclause}(a)$

THEOREM: find-contradiction-returns-proof-form
$((\neg\ \text{box-in-axioms}(x))$
$\wedge$ $(\text{excess-literal}(x) \simeq 0)$
$\wedge$ unsatisfiable (x)
$\wedge$ validclauses $(x))$
$\rightarrow$ proof-form $(\text{list}(\text{find-contradiction}(x)))$

THEOREM: proofs-together-have-form-if-both-do
$(\text{proof-form}\,(x) \wedge \text{proof-form}\,(y)) \rightarrow \text{proof-form}\,(\text{append}\,(x,\ y))$

THEOREM: adjust-does-not-unform-proofs
$(\text{proof-form}\,(x) \wedge \text{validliteral}\,(l)) \rightarrow \text{proof-form}\,(\text{adjust}\,(x,\ cs,\ l))$

```
;; getproof returns something that is of a resolution proof form
;; (later we show that this proof is "valid")
```

THEOREM: getproof-gives-proof-form
$(\text{unsatisfiable}\,(x) \wedge \text{validclauses}\,(x)) \rightarrow \text{proof-form}\,(\text{getproof}\,(x))$

THEOREM: allunits-means-no-excess-literal
$\text{allunits}\,(x) \rightarrow (\text{excess-literal}\,(x) \simeq \mathtt{0})$

THEOREM: find-contradiction-returns-box
$$\begin{array}{ll} \multicolumn{2}{l}{(\text{unsatisfiable}\,(x)} \\ \wedge & (\neg\ \text{box-in-axioms}\,(x)) \\ \wedge & (\text{excess-literal}\,(x) \simeq \mathtt{0}) \\ \wedge & \text{validclauses}\,(x)) \\ \rightarrow & (\text{car}\,(\text{find-contradiction}\,(x)) = \textbf{nil}) \end{array}$$

THEOREM: last-element-fact
$\text{listp}\,(x) \rightarrow (\text{last-element}\,(\text{append}\,(y,\ x)) = \text{last-element}\,(x))$

THEOREM: getproof-is-list-when
$(\neg\ \text{box-in-axioms}\,(x)) \rightarrow \text{listp}\,(\text{getproof}\,(x))$

```
;; the last resolvent in a getproof-generated proof is box
```

THEOREM: getproof-returns-box
$$\begin{array}{ll} \multicolumn{2}{l}{(\text{unsatisfiable}\,(x) \wedge \text{validclauses}\,(x) \wedge (\neg\ \text{box-in-axioms}\,(x)))} \\ \rightarrow & (\text{car}\,(\text{last-element}\,(\text{getproof}\,(x))) = \textbf{nil}) \end{array}$$

```
;; adjusting a proof ending in box yields a proof ending in box or a proof
;; whose last resolvent has just the literal used to adjust the proof
```

THEOREM: adjusted-proof-gets
$$\begin{array}{ll} \multicolumn{2}{l}{(\text{car}\,(\text{last-element}\,(p)) = \textbf{nil})} \\ \rightarrow & ((\text{car}\,(\text{last-element}\,(\text{adjust}\,(p,\ x,\ l))) = \textbf{nil}) \\ & \quad \vee \quad (\text{car}\,(\text{last-element}\,(\text{adjust}\,(p,\ x,\ l))) = \text{list}\,(l))) \end{array}$$

THEOREM: what-adjusted-proof-returns
$$\begin{array}{ll} \multicolumn{2}{l}{(\text{unsatisfiable}\,(x)} \\ \wedge & \text{validclauses}\,(x) \\ \wedge & (\neg\ \text{box-in-axioms}\,(x)) \\ \wedge & (\text{car}\,(\text{last-element}\,(\text{adjust}\,(\text{getproof}\,(x),\ y,\ l))) \neq \textbf{nil})) \\ \rightarrow & (\text{car}\,(\text{last-element}\,(\text{adjust}\,(\text{getproof}\,(x),\ y,\ l))) = \text{list}\,(l)) \end{array}$$

```
;; adjusting an "almost" correct proof yields a correct proof
```

THEOREM: adjusting-makes-proofs-valid

$$
\begin{array}{ll}
(\text{validliteral}\,(l) & \\
\ \wedge & \text{proof-form}\,(p) \\
\ \wedge & \text{validclauses}\,(cs) \\
\ \wedge & \text{validproof-but-for-literal}\,(cs,\ p,\ l)) \\
\rightarrow & \text{validproof}\,(cs,\ \text{adjust}\,(p,\ cs,\ l))
\end{array}
$$

THEOREM: validproof-of-append

$$
\begin{array}{ll}
(\text{proof-form}\,(x) & \\
\ \wedge & \text{proof-form}\,(y) \\
\ \wedge & \text{validclauses}\,(cs) \\
\ \wedge & \text{validproof}\,(cs,\ x) \\
\ \wedge & \text{validproof}\,(\text{append}\,(\text{listofcars}\,(x),\ cs),\ y)) \\
\rightarrow & \text{validproof}\,(cs,\ \text{append}\,(x,\ y))
\end{array}
$$

THEOREM: getproof-returns-validproof

$(\text{unsatisfiable}\,(x) \wedge \text{validclauses}\,(x)) \rightarrow \text{validproof}\,(x,\ \text{getproof}\,(x))$

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; the completeness theorem
```

THEOREM: resolution-is-complete

$(\text{unsatisfiable}\,(x) \wedge \text{validclauses}\,(x)) \rightarrow \text{finishedproof}\,(x,\ \text{getproof}\,(x))$

Index