Unit Resolution Proof Outline

Harry Bryant

April 2024

Contents

1 Key

- $l, l', l_0, l_1, l_2, l_3$ are Literals
- ls is a List of Literals that make up a Clause
- $\{l, ls\}$ is an example of a Clause - also referred to as c
- f is a Formula
- m is a Model

2 Unit Resolution implies Logical Entailment

2.1 Goal

If c can be derived from f with Unit Resolution, then c must be logically entailed from f.

Goal: $\textit{Unit Resolution}(f, c) \Rightarrow \textit{Logical Entailment}(f, c)$

2.2 Lemma

```
Lemma URes_implies_Entailment :
  forall (f : formula) (c : clause),
  unitres f c ->
  entails f c.
Proof.
```

2.3 Proof Description

1.	$\forall f, c\,(\text{unitres}(f,c) \Rightarrow \text{entails}(f,c))$	premise
2.	Assume unitres(f, c_1)	
3.	Induction on unitres(f, c_1) :	
4.	Subsumption Case:	
5.	Unfold entails	
6.	Assume model_property(m), models_formula(m, f)	
7.	Unfold models_formula	
8.	Specialize(H_2, H_1)	H_2: if c_1 is in f, then m models c_1, H_1: model property m
9.	Specialize(H_2, c, H)	H_2: if c_1 is in f, then m models c_1, c is a clause, H: c is a subset of c_1
10.	Unfold models_clause	
11.	Assume model_property(m)	
12.	Specialize(H_2, H_1)	H_2: if c_1 is in f, then m models c_1, H_1: model_property(m)
13.	Destruct H_2 as $(l_0\ \&\ (H_2L\ \&\ H_2R))$	H_2: there exists an l such that $l \in c$ and m models l, H_1: model_property(m)
14.	Let l_0 be a literal such that $l_0 \in c_1$ and $m \models l_0$	
15.	Apply l_0, H_2L, H_2R	H_2L: $l_0 \in c$, H_2R: m models l
16.	Therefore, entails(f, c_1)	
17.	Resolution Case:	
18.	By induction hypothesis,	
19.	entails(f, c)	
20.	entails$(f, \{\neg l\})$	
21.	Apply Lemma entailment_models (See Section 3) :	
22.	Apply IHU_1, IHU_2	IHU_1: entails(f, c), IHU_2: entails$(f, \{\neg l\})$
23.	Therefore, entails$(f$, remove_literal_from_clause$(l, c))$	
24.	Therefore, $\forall f, c\,(\text{unitres}(f,c) \Rightarrow \text{entails}(f,c))$	conclusion

3 Logical Entailment is Preserved

3.1 Goal

If c holds in f, and $\neg l$ holds in f, then c will still hold in f when l is removed

Goal: $(\mathit{Logical\ Entailment}(f, c) \wedge \mathit{Logical\ Entailment}(f, \{\neg l\})) \Rightarrow \mathit{Logical\ Entailment}(f, c \setminus l)$

3.2 Lemma

```
Lemma entailment_models :
  forall (f : formula) (c : clause) (l : literal),
    entails f c ->
    entails f [opposite l] ->
    entails f (remove_literal_from_clause l c).
Proof.
```

3.3 Proof Description

1.	$\forall f, c, l(\text{entails}(f, c) \Rightarrow \text{entails}(f, \{\neg l\})) \Rightarrow$ $\text{entails}(f, \text{remove\_literal\_from\_clause}(l, c)))$	premise
2.	Assume f, c, l	
3.	Assume $\text{entails}(f, c), \text{entails}(f, c \setminus l)$	
4.	Assume $m, \text{model\_property}(m)$	where m is a model
5.	Assume $\text{models\_formula}(m)$	
6.	(Show that m satisfies both c and $\neg l$)	
7.	Assert that $m \models c \wedge m \models \neg l$:	
8.	Case $m \models c$:	
9.	Apply $\text{entails}(f, c)$ to m	
10.	Therefore, $\text{models\_clause}(m, c)$	
11.	Case $m \models \neg l$:	
12.	Apply $\text{entails}(f, \neg l)$ to m	
13.	Therefore, $\text{models\_clause}(m, \{\neg l\})$	
14.	($\neg l$ being true in m implies l is false in m)	
15.	Unfold models_clause	in $\text{models\_clause}(m, \neg l)$
16.	Destruct Hm_neg_l as $(lit \mid (H\_mem \mid H\_model))$	Hm_neg_l: if m is a model, there exists a literal l_0 such that $l_0 \in c = \{\neg l\}$ and $m \models l_0$
17.	Apply $Hmodel\_prop$	$Hmodel\_prop$: $\text{model\_property}(m)$
18.	Destruct c :	
19.	Apply Lemma models_remove_literal_from_empty_clause: (See Section 11)	
20.	Apply Hm_c	Hm_c: $\text{models\_clause}(\emptyset)$
21.	Apply Lemma models_clause_remove_literal: (See Section 4)	
22.	Apply Hm_c, H	Hm_c: $\text{models\_clause}((l_0 :: c))$ H: $\text{models\_clause}(\{\neg l\})$
23.	Therefore, $\forall f, c, l(\text{entails}(f, c) \Rightarrow \text{entails}(f, \{\neg l\})) \Rightarrow$ $\text{entails}(f, \text{remove\_literal\_from\_clause}(l, c))$	conclusion

4 Modelling is Preserved

4.1 Goal

If m models c, and m models $\{\neg l\}$, then m will still model c when l is removed

Goal: $(Models(m, c) \wedge Models(m, c)) \Rightarrow Models(m, c \setminus l)$

4.2 Lemma

```
Lemma models_clause_remove_literal :
  forall (m : literal -> Prop) (c : clause) (l : literal),
    models_clause m c ->
    models_clause m [opposite l] ->
    models_clause m (remove_literal_from_clause l c).
Proof.
```

4.3 Proof Description

1.	$\forall m, c, l(\text{entails}(f, c) \Rightarrow \text{entails}(f, \{\neg l\})) \Rightarrow \text{entails}(f, c \setminus l)$	premise
2.	Assume m, c, l, Hm_c, Hm_neg_l	Hm_c: models_clause(m, c) Hm_neg_l: models_clause$(m, \{\neg l\})$
3.	Induction on c as $[\mid l' \, ls \, IHls]$.	$\mid$: empty list case l': head of the list ls: tail of the list $IHls$: Induction Hypothesis for the tail of the list
4.	Base case: c is empty	
5.	Rewrite Lemma remove_literal_from_empty_clause (See Section 11)	
6.	Apply Hm_c.	Hm_c: models_clause$(\emptyset)$
7.	Therefore, models_clause $(m, \emptyset \setminus l))$	
8.	Inductive case: $c = l' :: ls$	
9.	Destruct $(lit\_eq\_dec \, l' \, l)$ as $(Heq \mid Hneq)$.	
10.	Case 1: $l' = l$	
11.	See Section 4.3.1	
12.	Case 2: $l' \neq l$	
13.	See Section 4.3.2	
14.	Therefore, models_clause $(m, \{l', ls\} \setminus l)$	
15.	Therefore, $\forall m, c, l(\text{entails}(f, c) \Rightarrow \text{entails}(f, \{\neg l\})) \Rightarrow \text{entails}(f, c \setminus l)$	conclusion

4.3.1 Proof of Case 1: $l' = l$

1.	models_clause$(m, \{l', ls\} \setminus l)$	
2.	Rewrite Heq	$Heq : l' = l$
3.	Rewrite Lemma remove_l_from_cons_l (See Section 10)	
4.	Substitute l for l'	
5.	Apply $IHls$	$IHls$: if $m \models ls$, then $m \models ls \setminus l$
6.	Revert Hm_neg_l, Hm_c.	
7.	Apply Lemma $models\_ls$ (See Section 9) :	
8.	Therefore, models_clause$(m, \{l', ls\} \setminus l)$	

4.3.2 Proof of Case 2: $l' \neq l$

1.	models_clause$(m, \{l', ls\} \setminus l)$	
2.	Unfold models_clause	
3.	Assume models_property m	
4.	Apply Lemma models_c_implies_models_l_or_ls with m, l, l', ls, H, Hm_c as Hm_c' (See Section 8):	
5.	DestructHm_c'	Hm_c': models_clause$(m, \{l'\}) \vee$ models_clause(m, ls)
6.	Case $m \models \{l'\}$	
7.	See Proof Case 2A	
8.	Case $m \models ls$	
9.	See Proof Case 2B	
10.	Therefore, $\exists l_0 (l_0 \in \{l', ls\} \setminus l) \wedge (m \models l_0)$	

Proof of Case 2A: $m \models \{l'\}$

1.	$\exists l_0 (l_0 \in \{l', ls\} \setminus l) \wedge (m \models l_0)$	
2.	Apply Lemma rewrite_removal with $l, l', ls, Hneq$ as H_1 (See Section 7):	
3.	Rewrite H_1	$H_1 : \{l', ls\} \setminus l = \{l', ls \setminus l\}$
4.	Apply Lemma m_models_l_implies_m_models_l_colon_ls (See Section 5):	
5.	Apply H, H_0	H : model_property(m) H_0 : models_clause$(m, \{l'\})$
6.	Therefore, $\exists l_0 (l_0 \in \{l', ls\} \setminus l) \wedge (m \models l_0)$	

Proof of Case 2B: $m \models ls$

1.	$\exists l_0 (l_0 \in \{l', ls\} \setminus l) \wedge (m \models l_0)$	
2.	Specialize$(IHls, H_0)$	H : model_property(m) H_0 : models_clause$(m, \{l'\})$
3.	Apply Lemma rewrite_removal with $l, l', ls, Hneq$ as H_1 (See Section 7):	
4.	Rewrite H_1	$H_1 : \{l', ls\} \setminus l = \{l', ls \setminus l\}$
5.	Assert models_clause$(m, \{l', ls \setminus l\})$:	
6.	Apply m_models_ls_implies_m_models_l_colon_ls (See Section 6):	
7.	Apply H, IHls	H : model_property(m) H_0 : models_clause$(m, \{ls \setminus l\})$
8.		
9.	Unfold models_clause	
10.	Specialize(H_2, H)	H_2 : if m is a model, there exists a literal l_0 such that $l_0 \in \{l', ls \setminus l\}$ and $m \models l_0$ H : model_property(m)
11.	Destruct H_2 as $(l_0$ & $(H_2 l$ & $H_2 r))$:	
12.	Let l_0 be a literal such that $l_0 \in \{l', ls \setminus l\}$ and $m \models l_0$	
13.	Apply $H_2 l, H_2 r$	$H_2 l : \{l', ls \setminus l\}$ $H_2 r : m \models l_0$
14.	Therefore, $\exists l_0 (l_0 \in \{l', ls\} \setminus l) \wedge (m \models l_0)$	

5 If one element of a list is modelled, then the whole list is too

5.1 Goal

If m models l', then m will model the list $(l' :: ls)$

Goal: $(Models(m, \{l'\}) \Rightarrow Models(m, \{l', ls\})$

5.2 Lemma

```
Lemma m_models_l_implies_m_models_l_colon_ls :
  forall (m : literal -> Prop) (l' : literal) (ls : list literal),
  model_property m ->
  models_clause m [l'] ->
  models_clause m (l' :: ls).
Proof.
```

5.3 Proof Description

1.	$\forall m, l', ls$(model_property(m) $\Rightarrow$ models_clause(m, {l'}) $\Rightarrow$ models_clause(m, {l', ls}))	premise
2.	Assume m, l', ls, H, H_0	H: model_property(m) H_0: H_0: models_clause l'
3.	Unfold models_clause	
4.	Assume model_property(m)	
5.	Specialize (H_0, H_1)	
6.	Destruct H_0 as $(l_0 \;\&\; H_0')$:	
7.	Let l_0 be a literal such that $l_0 \in \{l', ls\}$ and $m \models l_0$	
8.	Split:	Now have: Goal 1: $l_0 \in \{l'\}$ Goal 2: $m \models l_0$
9.	Destruct H_0' as $(H_0'L \;\&\; H_0'R)$:	
10.	Unfold memlc, In	
11.	Left	
12.	Unfold memlc in $H_0'L$, In in $H_0'L$	$H_0'L : l_0 \in \{l'\}$ - becomes either $l = l_0 \vee False$
13.	Destruct $H0'L$ as $(H_0'LL \mid H_0'LR)$	
14.	Apply $H_0'LL$	$H_0'LL : l' = l_0$
15.	Contradiction	$False$ is an assumption
16.	Apply $H0'$	$H_0' : \cdots \wedge m \models l_0$
17.	Therefore, $\forall m, l', ls$(model_property(m) $\Rightarrow$ models_clause(m, {l'}) $\Rightarrow$ models_clause(m, {l', ls}))	conclusion

6 If one element in the tail of list is modelled, then the whole list is too

6.1 Goal

If m models an element in ls, then m will model the list $(l' :: ls)$

Goal: $(Models(m, \{ls'\}) \Rightarrow Models(m, \{l', ls\})$

6.2 Lemma

```
Lemma m_models_ls_implies_m_models_l_colon_ls :
  forall (m : literal -> Prop) (l' : literal) (ls : list literal),
  model_property m ->
  models_clause m ls ->
  models_clause m (l' :: ls).
Proof.
```

6.3 Proof Description

1.	$\forall m, l', ls$(model_property(m) $\Rightarrow$ models_clause(m, {ls'}) $\Rightarrow$ models_clause(m, {l', ls})	premise
2.	Assume m, l', ls, H, H_0	H: model_property(m) H_0: models_clause ls
3.	Unfold models_clause	
4.	Assume model_property(m)	
5.	Specialize (H_0, H_1)	
6.	Destruct H_0 as $(l_0 \;\&\; H_0')$:	
7.	Let l_0 be a literal such that $l_0 \in \{l', ls\}$ and $m \models l_0$	
8.	Split:	Now have: Goal 1: $l_0 \in \{l'\}$ Goal 2: $m \models l_0$
9.	Destruct H_0' as $(H_0'L \;\&\; H_0'R)$:	
10.	Unfold memlc, In	
11.	Right	
12.	Unfold memlc in $H_0'L$, In in $H_0'L$	$H_0'L : l_0 \in \{l'\}$ - becomes either $l = l_0 \vee (l_0 \in ls)$
13.	Apply $H_0'L, H0'R$	$H_0'L : l_0 \in ls$ $H_0'R : m \models l_0$
14.	Therefore, $\forall m, l', ls$(model_property(m) $\Rightarrow$ models_clause(m, {ls'}) $\Rightarrow$ models_clause(m, {l', ls})	conclusion

7 Rewriting List Removal when $l' \neq l$

7.1 Goal

If $l' \neq l$, removing l from the list $\{l', ls\}$ it is equivalent to $\{l', ls \setminus l\}$

Goal: $l' \neq l \Rightarrow \{l', ls\} \setminus l = \{l', ls \setminus l\}$

7.2 Lemma

```
Lemma rewrite_removal :
  forall (l l' : literal) (ls : list literal),
  l'<>l ->
  (remove_literal_from_clause l (l'::ls)) =
  (l' :: (remove lit_eq_dec l ls)).
Proof.
```

7.3 Proof Description

1.	$\forall l, l', ls(l' \neq l \Rightarrow (\{l', ls\} \setminus l) = (\{l', ls \setminus l\}))$	premise
2.	Assume l, l', ls, H_neq	$H\_neq : l' \neq l$
3.	Unfold remove_literal_from_clause	
4.	Simplify to reduce to literal definitions and statements	
5.	Assert $(l \neq l')$ as H_neq_rev	flip the $l' \neq l$
6.	Destruct (lit_eq_dec $l\ l'$) as $(H\_eq \mid H\_neq)$:	
7.	Case 1 $l = l'$:	
8.	Contradiction	
9.	Case 2 $l \neq l'$:	
10.	Reflexivity	
11.	Therefore, $\forall l, l', ls(l' \neq l \Rightarrow (\{l', ls\} \setminus l) = (\{l', ls \setminus l\}))$	conclusion

8 Modelling c implies l or ls is modelled

8.1 Goal

If m models the list $\{l', ls\}$, then it models either the head l' or the tail ls

Goal: $m \models \{l', ls\} \Rightarrow (m \models \{l\} \vee m \models \{ls\})$

8.2 Lemma

```
Lemma models_c_implies_models_l_or_ls:
forall (m : literal -> Prop) (l' : literal) (ls : list literal),
  model_property m ->
  models_clause m (l' :: ls) ->
  models_clause m [l'] \/ models_clause m ls.
Proof.
```

8.3 Proof Description

1.	$\forall m, l', ls(\text{model\_property}(m) \Rightarrow$ $\text{models\_clause}(m, \{l', ls\}) \Rightarrow$ $(\text{models\_clause }(m, \{l'\}) \vee \text{models\_clause}(m, \{ls\})))$	premise
2.	Assume m, l', ls, H, H_0	H: model_property m H_0: models_clause $m : (l' :: ls)$
3.	Destruct H_0	Now have: Goal 1: model_property(m) Goal 2: models_clause $(m, \{l'\}) \vee$ models_clause$(m, \{ls\})$
4.	Apply H	
5.	Destruct H_0 as $(Ha\ \&\ Hb)$	$Ha : x \in \{l', ls\}$ $Hb : m \models x$
6.	Destruct Ha as H_0	Now have: Goal 1: models_clause $(m, \{l'\}) \vee$ models_clause$(m, \{ls\})$ Goal 2: models_clause $(m, \{l'\}) \vee$ models_clause$(m, \{ls\})$
7.	Case $1 H_0 : l' = x :$	
8.	See Section 8.3.1	
9.	Case $2 H_0 : x \in ls :$	
10.	See Section 8.3.2	
11.	Therefore, $\forall m, l', ls(\text{model\_property}(m) \Rightarrow$ $\text{models\_clause}(m, \{l', ls\}) \Rightarrow$ $(\text{models\_clause }(m, \{l'\}) \vee \text{models\_clause}(m, \{ls\})))$	conclusion

8.3.1 Proof of Case 1: $H_0 : l' = x$

1.	models_clause$(m, \{l'\}) \vee$ models_clause$(m, \{ls\})$	
2.	Unfold models_clause	
3.	Left	Prove that a literal is modelled and is l', rather than in ls
4.	Let l' be a literal such that $l' \in \{l'\}$ and $m \models l'$	
5.	Split:	Now have: Goal 1: $l' \in \{l'\}$ Goal 2: $m \models l'$
6.	Unfold memlc	Becomes: $l' = l \vee False$
7.	Left	
8.	Reflexivity	
9.	Apply Hb	
10.	Therefore, models_clause$(m, \{l'\}) \vee$ models_clause$(m, \{ls\})$	

8.3.2 Proof of Case 2: $H_0 : x \in ls$

1.	models_clause$(m, \{l'\}) \vee$ models_clause$(m, \{ls\})$	
2.	Unfold models_clause	
3.	Right	Prove that a literal is modelled and member of ls, rather being l'
4.	Assume model_property(m)	
5.	Let x be a literal such that $x \in \{ls\}$ and $m \models x$	
6.	Split:	
7.	Apply H_0, Hb	
8.	Therefore, models_clause$(m, \{l'\}) \vee$ models_clause$(m, \{ls\})$	

9 Modelling c and $\neg l$ implies ls is modelled

9.1 Goal

If m models the list {l', ls}, and it models ¬l, then it must model ls

Goal: $(m \models \{l', ls\} \wedge m \models \{\neg l\}) \Rightarrow m \models \{ls\}$

9.2 Lemma

```
Lemma models_ls : forall
(m : literal -> Prop)
(ls : list literal)
(l : literal),
models_clause m (l :: ls) ->
models_clause m [opposite l] ->
models_clause m ls.
Proof.
```

9.3 Proof Description

1.	$\forall m, ls, l(\text{models\_clause}(m, \{l, ls\}) \Rightarrow$ $\text{models\_clause}(m, \{\neg l\}) \Rightarrow \text{models\_clause}(m, \{ls\})$	premise
2.	Assume m, ls, l	
3.	Unfold models_clause, memlc	Unfolding models_clause breaks it down to exists a literal which is in a list and is modelled Unfolding memlc breaks it down to $l_0 \in ls$
4.	Assume H, H_0, H_1	H : if m is a model, there exists a literal l_0 such that $l_0 \in \{l, ls\}$ and $m \models l_0$ H_0 : if m is a model, there exists a literal l_0 such that $l_0 \in \{\neg l\}$ and $m \models l_0$ H_1 : model_property(m)
5.	Specialize (H, H_1)	
6.	Specialize (H_0, H_1)	
7.	Destruct H as $(l_2\ \&\ (H_2L\ \&\ H_2R))$	
8.	Destruct H_0 as $(l_3\ \&\ (H_3L\ \&\ H_3R))$	
9.	Destruct H_2L	$l_2 \in \{l, ls\}$ Now have: Goal 1: $\exists l_0(l_0 \in \{ls\} \wedge m \models l_0)$ Goal 2: $\exists l_0(l_0 \in \{ls\} \wedge m \models l_0)$
10.	Case $l = l2$:	
11.	See Section 9.3.1	
12.	Case $l2$ in ls :	
13.	Let l_2 be a literal such that $l_2 \in \{ls\}$ and $m \models l_2$	
14.	Split:	Now have: Goal 1: $l_2 \in \{ls\}$ Goal 2: $m \models l_2$
15.	Apply H, H_2R	$H : l_2 \in \{ls\}$ $H_2R : m \models l_2$
16.	Therefore, $\exists l_0(l_0 \in \{ls\} \wedge m \models l_0)$	
17.	Therefore, $\forall m, ls, l(\text{models\_clause}(m, \{l, ls\}) \Rightarrow$ $\text{models\_clause}(m, \{\neg l\}) \Rightarrow \text{models\_clause}(m, \{ls\})$	conclusion

9.3.1 Proof of Case 1:

1.	$\exists l_0(l_0 \in \{ls\} \wedge m \models l_0)$	
2.	Replace l_2 with l in H_2R	
3.	Unfold model_property in H_1.	
4.	Assert $(m \models l \Rightarrow m \models \{\neg l\} \Rightarrow \text{ False})$:	
5.	Apply H_1	
6.	Specialize $(H0H2R)$	$H_0 : (m \models l \Rightarrow m \models \{\neg l\} \Rightarrow \text{ False}$ $H_2R : m \models l$
7.	Unfold In in H_3L	$H_3L : \neg l \in l_3$ becomes either $\neg l = l_3 \vee False$
8.	Destruct H_3L as $(H_3LL \mid H_3LR)$	
9.	Replace l_3 with l in H_3R	
10.	Contradiction, Contradiction	As $m \models l \wedge m \models \neg l$
11.	Therefore, $\exists l_0(l_0 \in \{ls\} \wedge m \models l_0)$	

10 Removing from a list

10.1 Goal

Removing l from a list $\{l', ls\}$ *is equivalent to removing l from ls*

Goal: $\{l', ls\} \setminus l \Rightarrow ls \setminus l$

10.2 Lemma

```
Lemma remove_l_from_cons_l : forall
(ls : list literal)
(l : literal), ((remove_literal_from_clause l (l :: ls)) =
(remove_literal_from_clause l ls)).
Proof.
```

10.3 Proof Description

1.	$\forall ls, l(\{l, ls\} \setminus l = \setminus l)$	premise
2.	Assume ls, l.	
3.	Simplify to reduce to literal definitions and statements	
4.	Destruct *lit_eq_dec l l* as $(H \mid H)$	
5.	Reflexivity	Goal is equivalent to the remove_literal_from_clause definition
6.	Contradiction	$l \neq l$ is an assumption
7.	Therefore, $\forall ls, l(\{l, ls\} \setminus l = \setminus l)$	conclusion
8.		

11 Modelling removing from an empty clause

11.1 Goal

If m models $\emptyset \setminus l$, then it will model $\emptyset$

Goal: $m \models (\emptyset \setminus l) \Leftrightarrow m \models \emptyset$

11.2 Lemma

```
Lemma models_remove_literal_from_empty_clause : forall m l,
  (models_clause m (remove_literal_from_clause l []) <->
  models_clause m []).
Proof.
```

11.3 Proof Description

1.	$\forall m, l, (\text{models\_clause}(m, \emptyset \setminus l)) \Leftrightarrow \text{models\_clause}(m, \emptyset))$	
2.	Assume m, l.	
3.	Split:	
4.	Case Forward Direction:	
5.	Assume models_clause$(m, (\emptyset \setminus l))$	
6.	Replace $\emptyset \setminus l$ with $\emptyset$ in H using Lemma remove_literal_from_empty_clause (See Section 12)	
7.	Apply H	H : models_clause$(m, \emptyset)$
8.	Therefore, models_clause$(m, \emptyset \setminus l) \Rightarrow$ *models_clause*$(m, \emptyset)$	
9.	Case Backward Direction:	
10.	Assume models_clause$(m, (\emptyset \setminus l))$	
11.	Replace $\emptyset \setminus l$ with $\emptyset$ using Lemma remove_literal_from_empty_clause (See Section 12)	
12.	Apply H	H : models_clause$(m, \emptyset)$
13.	Therefore, models_clause$(m, \emptyset) \Rightarrow$ *models_clause*$(m, \emptyset \setminus l)$	
14.	Therefore, $\forall m, l, (\text{models\_clause}(m, \emptyset \setminus l)) \Leftrightarrow \text{models\_clause}(m, \emptyset))$	conclusion

12 Removing from an empty clause

12.1 Goal

Removing l from a list $\{l', ls\}$ is equivalent to removing l from ls

Goal: $(\emptyset \setminus l) = \emptyset$

12.2 Lemma

```
Lemma remove_literal_from_empty_clause : forall l : literal ,
  remove_literal_from_clause l [] = [].
Proof.
```

12.3 Proof Description

1.	$\emptyset \setminus l = \emptyset$	premise
2.	Assume l	
3.	Unfold remove_literal_from_clause	Now, remove a literal if it equals one in the list
4.	Destruct l :	Now have: Goal 1: if positive literal = $\emptyset$ then remove it from $\emptyset$ Goal 2: if negative literal = $\emptyset$ then remove it from $\emptyset$
5.	Simplify, Reflexivity	
6.	Simplify, Reflexivity	
7.	Therefore, $\emptyset \setminus l = \emptyset$	conclusion

13 Unit Resolution preserves Falsity

13.1 Goal

If there is a model m and with Unit Resolution the Empty Clause can be derived, then there will not be a model for the Formula

Goal: $model\_property(m) \Rightarrow Unit\ Resolution(f, \emptyset) \Rightarrow \neg(m \models f)$

13.2 Lemma

```
Lemma unitres_no_model_false_formula :
  forall (m : literal -> Prop)
    (l : literal) (f : formula) (c : clause),
    model_property m ->
    unitres f [] ->
    models_formula m f ->
    False.
Proof.
```

13.3 Proof Description

1.	$\forall m, l, f, c$(model_property$(m) \Rightarrow$ unitres$(f, \emptyset) \Rightarrow$ models_formula$(m, f) \Rightarrow False)$	premise
2.	Assume $m, l, f, c,$ model_property, unitres$(f, \emptyset)$, models_formula(m, f)	
3.	Apply Lemma URes_implies_Entailment in H_0 (See Section 2)	H_0 was unitres$(f, \emptyset)$, becomes entails$(f, \emptyset)$
4.	Unfold entails in H_0	
5.	Specialize(H_0, m)	H_0: for any model m, if m is a model property, and $m \models f$, then $m \models \emptyset$ m : model_property m
6.	Apply H_0 in H	H_0: if m is a model property, and $m \models f$, then $m \models \emptyset$ H : model_property m
7.	Apply Lemma m_doesn't_model_falsity in H_0 (See Section 14)	Now have: Goal 1: $False$ Goal 2: $clause$ Goal 3: model_property m Goal 4: models_formula(m, f)
8.	Apply H, c, H_2, H_1	$H : False$ $c : clause$ H_2 : model_property m H_1 : models_formula(m, f)
9.	Therefore, $\forall m, l, f, c$(model_property$(m) \Rightarrow$ unitres$(f, \emptyset) \Rightarrow$ models_formula$(m, f) \Rightarrow False)$	conclusion

14 No Model for Falsity

14.1 Goal

There will be no model for the Empty Clause

Goal: $model\_property(m) \Rightarrow \neg(m \models \emptyset)$

14.2 Lemma

```
Lemma m_doesn't_model_falsity :
  forall (m : literal -> Prop) (c : clause),
  model_property m ->
  models_clause m [] ->
  False.
Proof.
```

14.3 Proof Description

1.	$\forall m, c(\text{model\_property}(m) \Rightarrow \text{ unitres}(f, \emptyset) \Rightarrow \text{models\_clause}(m, f))$	premise
2.	Assume $m, c, \text{model\_property}(m), \text{models\_clause}(m, \emptyset)$	
3.	unfold models_clause in H_0	
4.	specialize(H_0, H)	H_0: if m is a model property, then there exists a literal that is in the $\emptyset$ and is modelled by m H : model_property m
5.	Destruct H_0 as $(l_0 \,\&\, (H_0L \,\&\, H_0R))$	H_0 : there exists a literal that is in the $\emptyset$ and is modelled by m
6.	Contradiction	Because H_0L states $l_0 \in \emptyset$
7.	Therefore, $\forall m, c(\text{model\_property}(m) \Rightarrow \text{unitres}(f, \emptyset) \Rightarrow \text{ models\_clause}(m, f))$	conclusion