max planck institut
informatik

Variable and clause elimination for LTL satisfiability checking

Martin Suda

Max Planck Institut für Informatik

MACIS-2013

Linear temporal logic (LTL)

- modal logic for specifying temporal relations
- time modeled as a linear discrete sequence of time moments
- analysis of natural language expressibility (Kamp, 1968)
- specification language for systems with non-terminating computations (Pnueli, 1977)
  - model checking

Satisfiability checking of LTL formulas

- proving LTL theorems
- ensure quality of specifications
- LTL model checking reducible to LTL satisfiability

Linear temporal logic (LTL)

- modal logic for specifying temporal relations
- time modeled as a linear discrete sequence of time moments
- analysis of natural language expressibility (Kamp, 1968)
- specification language for systems with non-terminating computations (Pnueli, 1977)
  - model checking

Satisfiability checking of LTL formulas

- proving LTL theorems
- ensure quality of specifications
- LTL model checking reducible to LTL satisfiability

Linear temporal logic (LTL)

- modal logic for specifying temporal relations
- time modeled as a linear discrete sequence of time moments
- analysis of natural language expressibility (Kamp, 1968)
- specification language for systems with non-terminating computations (Pnueli, 1977)
  - model checking

Satisfiability checking of LTL formulas

- proving LTL theorems
- ensure quality of specifications
- LTL model checking reducible to LTL satisfiability

Linear temporal logic (LTL)

- modal logic for specifying temporal relations
- time modeled as a linear discrete sequence of time moments
- analysis of natural language expressibility (Kamp, 1968)
- specification language for systems with non-terminating computations (Pnueli, 1977)
  - model checking

Satisfiability checking of LTL formulas

- proving LTL theorems
- ensure quality of specifications
- LTL model checking reducible to LTL satisfiability

Linear temporal logic (LTL)

- modal logic for specifying temporal relations
- time modeled as a linear discrete sequence of time moments
- analysis of natural language expressibility (Kamp, 1968)
- specification language for systems with non-terminating computations (Pnueli, 1977)
  - model checking

Satisfiability checking of LTL formulas

- proving LTL theorems
- ensure quality of specifications
- LTL model checking reducible to LTL satisfiability

General resolution-based approach to satisfiability

- take the given formula φ
- translate it into a clausal normal form
  - clause: a disjunction of literals
  - literal: a variable or its negation
- derive new clauses by the resolution inference

$$\frac{C \vee p \quad D \vee \neg p}{C \vee D}$$

- until the empty clause $\perp$ is derived $\longrightarrow$ UNSAT
- or it is obvious this will not happen $\longrightarrow$ SAT
  - either by finding a model,
  - or by saturating the clause set

General resolution-based approach to satisfiability

- take the given formula φ
- translate it into a clausal normal form
  - clause: a disjunction of literals
  - literal: a variable or its negation
- derive new clauses by the resolution inference

$$\frac{C \vee p \quad D \vee \neg p}{C \vee D}$$

- until the empty clause $\perp$ is derived $\longrightarrow$ UNSAT
- or it is obvious this will not happen $\longrightarrow$ SAT
  - either by finding a model,
  - or by saturating the clause set

General resolution-based approach to satisfiability

- take the given formula φ
- translate it into a clausal normal form
  - clause: a disjunction of literals
  - literal: a variable or its negation
- derive new clauses by the resolution inference

$$\frac{C \vee p \quad D \vee \neg p}{C \vee D}$$

- until the empty clause $\perp$ is derived $\longrightarrow$ UNSAT
- or it is obvious this will not happen $\longrightarrow$ SAT
  - either by finding a model,
  - or by saturating the clause set

General resolution-based approach to satisfiability

- take the given formula φ
- translate it into a clausal normal form
  - clause: a disjunction of literals
  - literal: a variable or its negation
- derive new clauses by the resolution inference

$$\frac{C \vee p \quad D \vee \neg p}{C \vee D}$$

- until the empty clause $\perp$ is derived $\longrightarrow$ UNSAT
- or it is obvious this will not happen $\longrightarrow$ SAT
  - either by finding a model,
  - or by saturating the clause set

Preprocessing

- simplify the the normal form before starting the main algorithm
  1. removes redundancies of the original formula
  2. compensates for a potentially suboptimal NF-translation
- inspired by the SAT community:

Variable and clause elimination (Eén and Biere 2005)

- eliminate a variable by clause distribution
- remove tautologies (e.g., $C \vee p \vee \neg p$) and subsumed clauses ($C \subseteq D$)
- repeat while improving

Preprocessing

- simplify the the normal form before starting the main algorithm
  1. removes redundancies of the original formula
  2. compensates for a potentially suboptimal NF-translation
- inspired by the SAT community:

Variable and clause elimination (Eén and Biere 2005)

- eliminate a variable by clause distribution
- remove tautologies (e.g., $C \vee p \vee \neg p$) and subsumed clauses ($C \subseteq D$)
- repeat while improving

Preprocessing

- simplify the the normal form before starting the main algorithm
  1. removes redundancies of the original formula
  2. compensates for a potentially suboptimal NF-translation
- inspired by the SAT community:

Variable and clause elimination (Eén and Biere 2005)

- eliminate a variable by clause distribution
- remove tautologies (e.g., $C \vee p \vee \neg p$) and subsumed clauses ($C \subseteq D$)
- repeat while improving

Propositional variable elimination (by clause distribution)

- "Rule for Eliminating Atomic Formulas"
  (Davis and Putnam 1960)
- given a variable p, separate clause set N based on p

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

- distribute over p

$$N_p \otimes N_{\neg p} = \{(C \vee D) \mid (C \vee p) \in N_p, (D \vee \neg p) \in N_{\neg p}\}$$

- replace N_p and $N_{\neg p}$ in N by the result

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

- p no longer occurs; the set is equisatisfiable

Propositional variable elimination (by clause distribution)

- "Rule for Eliminating Atomic Formulas" (Davis and Putnam 1960)
- given a variable *p*, separate clause set *N* based on *p*

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

- distribute over *p*

$$N_p \otimes N_{\neg p} = \{(C \vee D) \mid (C \vee p) \in N_p, (D \vee \neg p) \in N_{\neg p}\}$$

- replace N_p and $N_{\neg p}$ in *N* by the result

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

- *p* no longer occurs; the set is equisatisfiable

Propositional variable elimination (by clause distribution)

- “Rule for Eliminating Atomic Formulas”
  (Davis and Putnam 1960)
- given a variable p, separate clause set N based on p

$$N = N_p \,\dot\cup\, N_{\neg p} \,\dot\cup\, N_0$$

- distribute over p

$$N_p \otimes N_{\neg p} = \{(C \vee D) \mid (C \vee p) \in N_p, (D \vee \neg p) \in N_{\neg p}\}$$

- replace N_p and $N_{\neg p}$ in N by the result

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

- p no longer occurs; the set is equisatisfiable

Propositional variable elimination (by clause distribution)

- "Rule for Eliminating Atomic Formulas" (Davis and Putnam 1960)
- given a variable p, separate clause set N based on p

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

- distribute over p

$$N_p \otimes N_{\neg p} = \{(C \vee D) \mid (C \vee p) \in N_p, (D \vee \neg p) \in N_{\neg p}\}$$

- replace N_p and $N_{\neg p}$ in N by the result

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

- p no longer occurs; the set is equisatisfiable

Propositional variable elimination (by clause distribution)

- "Rule for Eliminating Atomic Formulas" (Davis and Putnam 1960)
- given a variable p, separate clause set N based on p

$$N = N_p \,\dot{\cup}\, N_{\neg p} \,\dot{\cup}\, N_0$$

- distribute over p

$$N_p \otimes N_{\neg p} = \{(C \vee D) \mid (C \vee p) \in N_p, (D \vee \neg p) \in N_{\neg p}\}$$

- replace N_p and $N_{\neg p}$ in N by the result

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

- p no longer occurs; the set is equisatisfiable

The main challenge of preprocessing in LTL

- the normal form consists of temporal clauses
  - bound to a specific temporal context
  - interactions need to be controlled
- one variable may refer to more than one time point

Solution proposed by this work

- further refine the traditional normal form
- assign labels to clauses to track their temporal relations
- enables us to “lift” resolution-based reasoning from SAT to LTL
- and, in particular, to lift variable and clause elimination

The main challenge of preprocessing in LTL

- the normal form consists of temporal clauses
  - bound to a specific temporal context
  - interactions need to be controlled
- one variable may refer to more than one time point

Solution proposed by this work

- further refine the traditional normal form
- assign labels to clauses to track their temporal relations
- enables us to “lift” resolution-based reasoning from SAT to LTL
- and, in particular, to lift variable and clause elimination

The main challenge of preprocessing in LTL

- the normal form consists of temporal clauses
  - bound to a specific temporal context
  - interactions need to be controlled
- one variable may refer to more than one time point

Solution proposed by this work

- further refine the traditional normal form
- assign labels to clauses to track their temporal relations
- enables us to "lift" resolution-based reasoning from SAT to LTL
- and, in particular, to lift variable and clause elimination

The main challenge of preprocessing in LTL

- the normal form consists of temporal clauses
  - bound to a specific temporal context
  - interactions need to be controlled
- one variable may refer to more than one time point

Solution proposed by this work

- further refine the traditional normal form
- assign labels to clauses to track their temporal relations
- enables us to “lift” resolution-based reasoning from SAT to LTL
- and, in particular, to lift variable and clause elimination

LTL primer

- basic signature: $\Sigma = \{p, q, \dots\}$
- prop. logic syntax plus: next $\bigcirc$, always $\Box$, sometime $\Diamond$, ...
- prop. valuation a.k.a. state: $W : \Sigma \to \{0, 1\}$
- LTL interpretation – a sequence of states: $\mathcal{W} = (W_i)_{i \in \mathbb{N}}$

Semantics

$\mathcal{W}, i \models p$	iff $W_i \models p$,
$\mathcal{W}, i \models \neg\varphi$	iff not $\mathcal{W}, i \models \varphi$,
$\mathcal{W}, i \models \varphi \wedge (\vee) \psi$	iff $\mathcal{W}, i \models \varphi$ and (or) $\mathcal{W}, i \models \psi$,
$\mathcal{W}, i \models \bigcirc\varphi$	iff $\mathcal{W}, i+1 \models \varphi$,
$\mathcal{W}, i \models \Box\varphi$	iff for every $j \geq i$, $\mathcal{W}, j \models \varphi$,
$\mathcal{W}, i \models \Diamond\varphi$	iff for some $j \geq i$, $\mathcal{W}, j \models \varphi$,
...	

LTL primer

- basic signature: $\Sigma = \{p, q, \ldots\}$
- prop. logic syntax plus: next $\bigcirc$, always $\Box$, sometime $\Diamond$, ...
- prop. valuation a.k.a. state: $W : \Sigma \to \{0, 1\}$
- LTL interpretation – a sequence of states: $\mathcal{W} = (W_i)_{i \in \mathbb{N}}$

Semantics

$$
\begin{array}{ll}
\mathcal{W}, i \models p & \text{iff } W_i \models p, \\
\mathcal{W}, i \models \neg\varphi & \text{iff not } \mathcal{W}, i \models \varphi, \\
\mathcal{W}, i \models \varphi \wedge (\vee) \psi & \text{iff } \mathcal{W}, i \models \varphi \text{ and (or) } \mathcal{W}, i \models \psi, \\
\mathcal{W}, i \models \bigcirc\varphi & \text{iff } \mathcal{W}, i+1 \models \varphi, \\
\mathcal{W}, i \models \Box\varphi & \text{iff for every } j \geq i, \mathcal{W}, j \models \varphi, \\
\mathcal{W}, i \models \Diamond\varphi & \text{iff for some } j \geq i, \mathcal{W}, j \models \varphi, \\
\ldots &
\end{array}
$$

Separated Normal Form (Fisher 1991) for an LTL formula

$$
\begin{array}{rcl}
\varphi & \longrightarrow & \mathbf{i} \wedge \tau[\square(\neg \mathbf{i} \vee \varphi)], \\
\\
\tau[\square(\neg x \vee l)] & \longrightarrow & \square(\neg x \vee l), \text{ if } l \text{ is a literal}, \\
\tau[\square(\neg x \vee (\varphi \wedge \psi))] & \longrightarrow & \tau[\square(\neg x \vee \varphi)] \wedge \tau[\square(\neg x \vee \psi)], \\
\tau[\square(\neg x \vee (\varphi \vee \psi))] & \longrightarrow & \square(\neg x \vee \mathbf{u} \vee \mathbf{v}) \wedge \\
 & & \tau[\square(\neg \mathbf{u} \vee \varphi)] \wedge \tau[\square(\neg \mathbf{v} \vee \psi)], \\
\tau[\square(\neg x \vee \bigcirc \varphi)] & \longrightarrow & \square(\neg x \vee \bigcirc \mathbf{u}) \wedge \tau[\square(\neg \mathbf{u} \vee \varphi)], \\
\tau[\square(\neg x \vee \square \varphi)] & \longrightarrow & \square(\neg x \vee \mathbf{u}), \ \wedge \\
 & & \square(\neg \mathbf{u} \vee \bigcirc \mathbf{u}) \wedge \tau[\square(\neg \mathbf{u} \vee \varphi)], \\
\tau[\square(\neg x \vee \diamond \varphi)] & \longrightarrow & \square(\neg x \vee \diamond \mathbf{u}) \wedge \tau[\square(\neg \mathbf{u} \vee \varphi)], \\
 & \ldots &
\end{array}
$$

Temporal Satisfiability Task (TST)

- further refine SNF (Degtyarev et al. 2002)
- use priming notation to denote next ($\bigcirc p \longrightarrow p'$)
- Initial clauses I, step clauses T, and goal clauses G

$$\left(\bigwedge_{C_i \in I} C_i\right) \wedge \square \left(\bigwedge_{C_t \vee D_t' \in T} (C_t \vee \bigcirc D_t)\right) \wedge \square\Diamond \left(\bigwedge_{C_g \in G} C_g\right)$$

Semantics in a picture

$\Sigma_0 \quad \Sigma_1 \quad \Sigma_2 \quad \cdots$

Temporal Satisfiability Task (TST)

- further refine SNF (Degtyarev et al. 2002)
- use priming notation to denote next ($\bigcirc p \longrightarrow p'$)
- Initial clauses I, step clauses T, and goal clauses G

$$\left(\bigwedge_{C_i \in I} C_i\right) \wedge \Box \left(\bigwedge_{C_t \vee D_t' \in T} (C_t \vee \bigcirc D_t)\right) \wedge \Box\Diamond \left(\bigwedge_{C_g \in G} C_g\right)$$

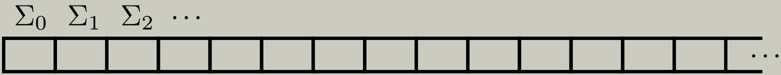
Semantics in a picture

Temporal Satisfiability Task (TST)

- further refine SNF (Degtyarev et al. 2002)
- use priming notation to denote next ($\bigcirc p \longrightarrow p'$)
- Initial clauses I, step clauses T, and goal clauses G

$$\left(\bigwedge_{C_i \in I} C_i\right) \wedge \Box \left(\bigwedge_{C_t \vee D_t' \in T} (C_t \vee \bigcirc D_t)\right) \wedge \Box\Diamond \left(\bigwedge_{C_g \in G} C_g\right)$$

Semantics in a picture

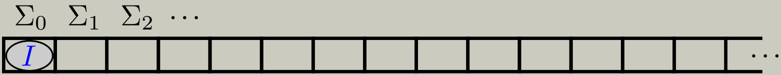


Temporal Satisfiability Task (TST)

- further refine SNF (Degtyarev et al. 2002)
- use priming notation to denote next ($\bigcirc p \longrightarrow p'$)
- Initial clauses I, step clauses T, and goal clauses G

$$\left(\bigwedge_{C_i \in I} C_i\right) \wedge \Box \left(\bigwedge_{C_t \vee D_t' \in T} (C_t \vee \bigcirc D_t)\right) \wedge \Box\Diamond \left(\bigwedge_{C_g \in G} C_g\right)$$

Semantics in a picture

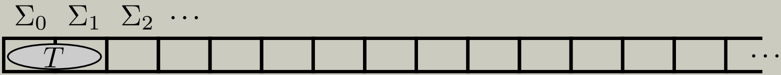


Temporal Satisfiability Task (TST)

- further refine SNF (Degtyarev et al. 2002)
- use priming notation to denote next ($\bigcirc p \quad \longrightarrow \quad p'$)
- Initial clauses I, step clauses T, and goal clauses G

$$\left(\bigwedge_{C_i \in I} C_i\right) \wedge \Box \left(\bigwedge_{C_t \vee D_t' \in T} (C_t \vee \bigcirc D_t)\right) \wedge \Box\Diamond \left(\bigwedge_{C_g \in G} C_g\right)$$

Semantics in a picture

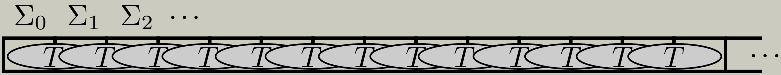


Temporal Satisfiability Task (TST)

- further refine SNF (Degtyarev et al. 2002)
- use priming notation to denote next ($\bigcirc p \quad \longrightarrow \quad p'$)
- Initial clauses I, step clauses T, and goal clauses G

$$\left(\bigwedge_{C_i \in I} C_i\right) \wedge \Box \left(\bigwedge_{C_t \vee D_t' \in T} (C_t \vee \bigcirc D_t)\right) \wedge \Box\Diamond \left(\bigwedge_{C_g \in G} C_g\right)$$

Semantics in a picture

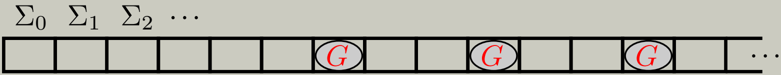


Temporal Satisfiability Task (TST)

- further refine SNF (Degtyarev et al. 2002)
- use priming notation to denote next ($\bigcirc p \longrightarrow p'$)
- Initial clauses I, step clauses T, and goal clauses G

$$\left(\bigwedge_{C_i \in I} C_i\right) \wedge \Box \left(\bigwedge_{C_t \vee D_t' \in T} (C_t \vee \bigcirc D_t)\right) \wedge \Box\Diamond \left(\bigwedge_{C_g \in G} C_g\right)$$

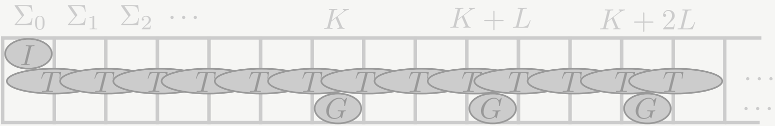
Semantics in a picture



(K, L)-models

- We can assume the time indexes of the *G*-states form an arithmetic progression $j = K + i \cdot L$ for some $K \in \mathbb{N}$ and $L \in \mathbb{N}^+$

Reducing to propositional logic

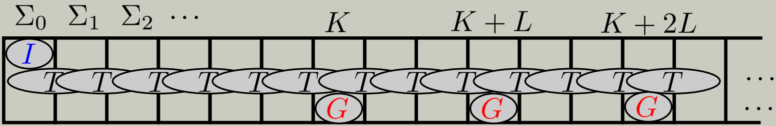


- Once the placement of the *G*-states is fixed, we are left with an infinite set of standard clauses over an infinite signature.
- It is just copies of the original clauses shifted in time ...

(K, L)-models

- We can assume the time indexes of the G-states form an arithmetic progression $j = K + i \cdot L$ for some $K \in \mathbb{N}$ and $L \in \mathbb{N}^+$

Reducing to propositional logic



- Once the placement of the G-states is fixed, we are left with an infinite set of standard clauses over an infinite signature.
- It is just copies of the original clauses shifted in time ...

“Lifting” with labels

We annotate the original clauses with labels in order to

- finitely represent the infinite set of clauses,
- reason about all possible G-state placements at once.

Starting label assignment

initial I	$\longrightarrow$	$\bigwedge C_i$	$\longrightarrow$	$\bigwedge(0,*,0)\|\|C_i$
step T	$\longrightarrow$	$\bigwedge C_t$	$\longrightarrow$	$\bigwedge(*,*,0)\|\|C_t$
goal G	$\longrightarrow$	$\bigwedge C_g$	$\longrightarrow$	$\bigwedge(*,0,0)\|\|C_g$

"Lifting" with labels

We annotate the original clauses with labels in order to

- finitely represent the infinite set of clauses,
- reason about all possible G-state placements at once.

Starting label assignment

$$
\begin{array}{lclcl}
\text{initial } I & \longrightarrow & \bigwedge C_i & \longrightarrow & \bigwedge(0,*,0)||C_i \\
\text{step } T & \longrightarrow & \bigwedge C_t & \longrightarrow & \bigwedge(*,*,0)||C_t \\
\text{goal } G & \longrightarrow & \bigwedge C_g & \longrightarrow & \bigwedge(*,0,0)||C_g
\end{array}
$$

Labeled resolution

$$\mathcal{I}\frac{(b_1, k_1, l_1) \mid\mid C_1 \vee p \qquad (b_2, k_2, l_2) \mid\mid C_2 \vee \neg p}{(b, k, l) \mid\mid C \vee D}$$

- where (b, k, l) is the merge of labels (b_1, k_1, l_1) and (b_2, k_2, l_2)
  - intuitively captures intersection of the represented contexts
- up to infinitely many prop. resolutions correspond to one labeled inference

Temporal shift

- need to align unprimed and primed symbols in labeled clauses
- we prefix resolution with a shift of one of the premises

Labeled resolution

$$\mathcal{I}\frac{(b_1, k_1, l_1) \mid\mid C_1 \vee p \qquad (b_2, k_2, l_2) \mid\mid C_2 \vee \neg p}{(b, k, l) \mid\mid C \vee D}$$

- where (b, k, l) is the merge of labels (b_1, k_1, l_1) and (b_2, k_2, l_2)
  - intuitively captures intersection of the represented contexts
- up to infinitely many prop. resolutions correspond to one labeled inference

Temporal shift

- need to align unprimed and primed symbols in labeled clauses
- we prefix resolution with a shift of one of the premises

Example

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

Example

$$N = N_p \,\dot\cup\, N_{\neg p} \,\dot\cup\, N_0$$

$$\begin{array}{ll}
(0,*,0) \,||\, p \vee q \vee r & (*,0,0) \,||\, \neg p \vee q \\
(*,0,0) \,||\, p \vee \neg q & (0,*,0) \,||\, \neg p \vee \neg r \\
 & (*,*,0) \,||\, r \vee \neg p'
\end{array}$$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

Example

$$N = N_p \,\dot\cup\, N_{\neg p} \,\dot\cup\, N_0$$

$(0,*,0)\,\|\|\,p \vee q \vee r$	$(*,0,0)\,\|\|\,\neg p \vee q$	$(0,0,0)\,\|\|\,q \vee r$
$(*,0,0)\,\|\|\,p \vee \neg q$	$(0,*,0)\,\|\|\,\neg p \vee \neg r$	
	$(*,*,0)\,\|\|\,r \vee \neg p'$	

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

Example

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

$(0,*,0) \,||\, p \vee q \vee r$ $\quad$ $(*,0,0) \,||\, \neg p \vee q$

$(*,0,0) \,||\, p \vee \neg q$ $\quad$ $(0,*,0) \,||\, \neg p \vee \neg r$

$(*,*,0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0,0,0) \,||\, q \vee r$

Example

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$ $\quad$ $(*, 0, 0) \,||\, \neg p \vee q$

$(*, 0, 0) \,||\, p \vee \neg q$ $\quad$ $(0, *, 0) \,||\, \neg p \vee \neg r$ $\quad$ $(0, *, 0) \,||\, q \vee r \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0, 0, 0) \,||\, q \vee r$

Example

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$ $\quad$ $(*, 0, 0) \,||\, \neg p \vee q$

$(*, 0, 0) \,||\, p \vee \neg q$ $\quad$ $(0, *, 0) \,||\, \neg p \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0, 0, 0) \,||\, q \vee r$

Example

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$ $\quad$ $(*, 0, 0) \,||\, \neg p \vee q$

$(*, 0, 0) \,||\, p \vee \neg q$ $\quad$ $(0, *, 0) \,||\, \neg p \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$ $\quad \bot$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0, 0, 0) \,||\, q \vee r$

Example

$$N = N_p \,\dot{\cup}\, N_{\neg p} \,\dot{\cup}\, N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$ $\quad$ $(*, 0, 0) \,||\, \neg p \vee q$

$(*, 0, 0) \,||\, p \vee \neg q$ $\quad$ $(0, *, 0) \,||\, \neg p \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0, 0, 0) \,||\, q \vee r$

Example

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$ $\quad (*, 0, 0) \,||\, \neg p \vee q \quad (*, 0, 0) \,||\, q \vee \neg q$

$(*, 0, 0) \,||\, p \vee \neg q$ $\quad (0, *, 0) \,||\, \neg p \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0, 0, 0) \,||\, q \vee r$

Example

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$

$(*, 0, 0) \,||\, p \vee \neg q$

$(*, 0, 0) \,||\, \neg p \vee q$

$(0, *, 0) \,||\, \neg p \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0, 0, 0) \,||\, q \vee r$

Example

$$N = N_p \,\dot\cup\, N_{\neg p} \,\dot\cup\, N_0$$

$(0,*,0) \,||\, p \vee q \vee r$ $\quad$ $(*,0,0) \,||\, \neg p \vee q$

$(*,0,0) \,||\, p \vee \neg q$ $\quad$ $(0,*,0) \,||\, \neg p \vee \neg r$ $\quad$ $(0,0,0) \,||\, \neg q \vee \neg r$

$(*,*,0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0,0,0) \,||\, q \vee r$

Example

$$N = N_p \,\dot\cup\, N_{\neg p} \,\dot\cup\, N_0$$

$(0,*,0) \,||\, p \vee q \vee r$

$(*,0,0) \,||\, p \vee \neg q$

$(*,0,0) \,||\, \neg p \vee q$

$(0,*,0) \,||\, \neg p \vee \neg r$

$(*,*,0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0,0,0) \,||\, q \vee r$

$(0,0,0) \,||\, \neg q \vee \neg r$

Example

$$N = N_p \,\dot\cup\, N_{\neg p} \,\dot\cup\, N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$

$(*, 0, 0) \,||\, p \vee \neg q$

$(*, 0, 0) \,||\, \neg p \vee q$

$(0, *, 0) \,||\, \neg p \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0, 0, 0) \,||\, q \vee r$

$(0, 0, 0) \,||\, \neg q \vee \neg r$

Example

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$

$(*, 1, 0) \,||\, p' \vee \neg q'$

$(*, 0, 0) \,||\, \neg p \vee q$

$(0, *, 0) \,||\, \neg p \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0, 0, 0) \,||\, q \vee r$

$(0, 0, 0) \,||\, \neg q \vee \neg r$

Example

$$N = N_p \dot{\cup} N_{\neg p} \dot{\cup} N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$

$(*, 1, 0) \,||\, p' \vee \neg q'$

$(*, 0, 0) \,||\, \neg p \vee q$

$(0, *, 0) \,||\, \neg p \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$ $\quad$ $(*, 1, 0) \,||\, r \vee \neg q'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0, 0, 0) \,||\, q \vee r$

$(0, 0, 0) \,||\, \neg q \vee \neg r$

Example

$$N = N_p \dot{\cup} N_{\neg p} \dot{\cup} N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$

$(*, 0, 0) \,||\, p \vee \neg q$

$(*, 0, 0) \,||\, \neg p \vee q$

$(0, *, 0) \,||\, \neg p \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(0, 0, 0) \,||\, q \vee r$

$(0, 0, 0) \,||\, \neg q \vee \neg r$

$(*, 1, 0) \,||\, r \vee \neg q'$

Example

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

$(0,*,0) \,||\, p \vee q \vee r$

$(*,0,0) \,||\, p \vee \neg q$

$(*,0,0) \,||\, \neg p \vee q$

$(0,*,0) \,||\, \neg p \vee \neg r$

$(*,*,0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$\cancel{(0,0,0) \,||\, q \vee r}$

$\cancel{(0,0,0) \,||\, \neg q \vee \neg r}$

$(*,1,0) \,||\, r \vee \neg q'$

Example

$$N = N_p \mathbin{\dot\cup} N_{\neg p} \mathbin{\dot\cup} N_0$$

$(0, *, 0) \,||\, p \vee q \vee r$

$(*, 0, 0) \,||\, p \vee \neg q$

$(*, 0, 0) \,||\, \neg p \vee q$

$(0, *, 0) \,||\, \neg p \vee \neg r$

$(*, *, 0) \,||\, r \vee \neg p'$

$$\overline{N} = (N_p \otimes N_{\neg p}) \cup N_0$$

$(*, 1, 0) \,||\, r \vee \neg q'$

Limitations

- cannot eliminate variables occurring both primed and unprimed

$$p \vee q \vee p' \vee \neg r'$$

  (the result may not be expressible in LTL)

- clauses with multiple primes are meaningful but obtrusive

$$\frac{p \vee r' \qquad \neg r \vee \neg q'}{p \vee \neg q''}$$

  (no problem if later shown redundant)

Limitations

- cannot eliminate variables occurring both primed and unprimed

$$p \vee q \vee p' \vee \neg r'$$

  (the result may not be expressible in LTL)

- clauses with multiple primes are meaningful but obtrusive

$$\frac{p \vee r' \quad \neg r \vee \neg q'}{p \vee \neg q''}$$

  (no problem if later shown redundant)

Prototype implementation based on Minisat 2.2

- reuse the SAT solver's simplification loop
- emulate labels by marking literals

Input problems

- 3723 formulas collected by Schuppan and Darmawan (2011)
- several families, various flavors (application, crafted, random)

Two resolution LTL provers

- LS4: an LTL prover with partial model guidance (Suda and Wiedenbach, 2012)
- trp++: saturation prover using CTR (Hustadt and Konev, 2003)

Prototype implementation based on Minisat 2.2

- reuse the SAT solver's simplification loop
- emulate labels by marking literals

Input problems

- 3723 formulas collected by Schuppan and Darmawan (2011)
- several families, various flavors (application, crafted, random)

Two resolution LTL provers

- LS4: an LTL prover with partial model guidance (Suda and Wiedenbach, 2012)
- trp++: saturation prover using CTR (Hustadt and Konev, 2003)

Prototype implementation based on Minisat 2.2

- reuse the SAT solver's simplification loop
- emulate labels by marking literals

Input problems

- 3723 formulas collected by Schuppan and Darmawan (2011)
- several families, various flavors (application, crafted, random)

Two resolution LTL provers

- LS4: an LTL prover with partial model guidance (Suda and Wiedenbach, 2012)
- trp++: saturation prover using CTR (Hustadt and Konev, 2003)

Phase 1: translation

- Of the original formulas (general LTL) ...
- ...to TST's (accessible to both provers)

Phase 2: simplification

- recording number of variables and clauses eliminated
- in total: 39 % of the variables (7% original, 32% auxiliary) and 32 % of clauses eliminated
- numbers vary across the individual families

Phase 3: effect of simplification on prover runtime

- attempt solving original and simplified version of the problem
- 300 second time limit per problem

Phase 1: translation

- Of the original formulas (general LTL) ...
- ...to TST's (accessible to both provers)

Phase 2: simplification

- recording number of variables and clauses eliminated
- in total: 39 % of the variables (7% original, 32% auxiliary) and 32 % of clauses eliminated
- numbers vary across the individual families

Phase 3: effect of simplification on prover runtime

- attempt solving original and simplified version of the problem
- 300 second time limit per problem

Phase 1: translation

- Of the original formulas (general LTL) ...
- ... to TST's (accessible to both provers)

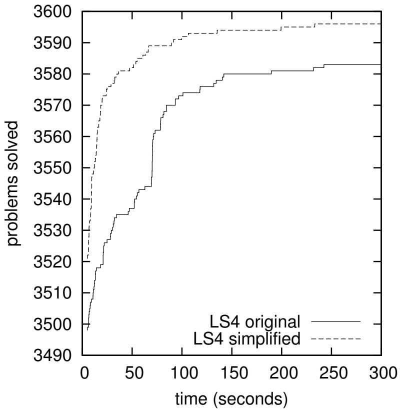
Phase 2: simplification

- recording number of variables and clauses eliminated
- in total: 39 % of the variables (7% original, 32% auxiliary) and 32 % of clauses eliminated
- numbers vary across the individual families

Phase 3: effect of simplification on prover runtime

- attempt solving original and simplified version of the problem
- 300 second time limit per problem

family	size		LS4		trp++	
			solved	time	solved	time
`acacia`	71	o	71	7.1s	71	39.3s
		s	71	7.1s	71	11.3s
`alaska`	140	o	121	6607.0s	9	39423.2s
		s	139	882.0s	12	38717.5s
`anzu`	111	o	93	5754.2s	0	33300.0s
		s	94	5482.2s	0	33300.0s
`forobots`	39	o	39	4.3s	39	1198.8s
		s	39	3.9s	39	194.2s
`rozier`	2320	o	2278	13312.9s	2063	96293.7s
		s	2278	13270.7s	2120	76921.1s
`schuppan`	72	o	41	9332.8s	36	11189.8s
		s	41	9320.9s	37	10741.0s
`trp`	970	o	940	12327.5s	364	189045.2s
		s	934	11887.5s	359	190138.3s
total	3723	o	3583	47345.8s	2582	370490.0s
		s	3596	40854.3s	2638	350023.4s



3600
3590
3580
3570
3560
3550
3540
3530
3520
3510
3500
3490
problems solved
0 50 100 150 200 250 300
time (seconds)
LS4 original
LS4 simplified
2650
2600
2550
2500
2450
2400
2350
2300
2250
2200
2150
problems solved
0 50 100 150 200 250 300
time (seconds)
trp++ original
trp++ simplified

Summary

- a new preprocessing technique for LTL satisfiability
- mechanism of labeled clauses effectively "lifts" variable and clause elimination from SAT to LTL
- could other techniques be generalized as well?
  - e.g., blocked clause elimination (Järvisalo et al. 2010)?

Summary

- a new preprocessing technique for LTL satisfiability
- mechanism of labeled clauses effectively “lifts” variable and clause elimination from SAT to LTL
- could other techniques be generalized as well?
  - e.g., blocked clause elimination (Järvisalo et al. 2010)?

Summary

- a new preprocessing technique for LTL satisfiability
- mechanism of labeled clauses effectively “lifts” variable and clause elimination from SAT to LTL
- could other techniques be generalized as well?
  - e.g., blocked clause elimination (Järvisalo et al. 2010)?