Resolution-based Methods for Linear Temporal Reasoning

– PhD dissertation defense –

Martin Suda

Saarbrücken, October 16, 2015

Linear Time Reasoning

- reasoning about systems that evolve in time
- $\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \cdots$
- model = sequence of *propositional* interpretations, “worlds”

Applications

- reactive systems: protocols, hardware circuits, . . .
- automated planning
- dynamic authorization policies, . . .

Characteristics

- temporal aspect increases complexity from NP to PSPACE
- exponential model / inductive argument

Linear Time Reasoning

- reasoning about systems that evolve in time
- $\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \cdots$
- model = sequence of *propositional* interpretations, “worlds”

Applications

- reactive systems: protocols, hardware circuits, . . .
- automated planning
- dynamic authorization policies, . . .

Characteristics

- temporal aspect increases complexity from NP to PSPACE
- exponential model / inductive argument

Linear Time Reasoning

- reasoning about systems that evolve in time
- $\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \cdots$
- model = sequence of *propositional* interpretations, “worlds”

Applications

- reactive systems: protocols, hardware circuits, . . .
- automated planning
- dynamic authorization policies, . . .

Characteristics

- temporal aspect increases complexity from NP to PSPACE
- exponential model / inductive argument

Resolution-based Methods

- resolution [Davis and Putnam, 1960]

$$\mathcal{I}\,\frac{C \vee a \quad D \vee \neg a}{C \vee D}$$

- superposition [Bachmair and Ganzinger, 1990, 1994]
  - equality rule + completeness argument
  - nice theoretical properties
  - foundation for successful implementations
- modern SAT solving
  - DPLL [Davis et al., 1962]
  - CDCL [Marques-Silva and Sakallah, 1999]
  - backtrack search + implicit resolution

Five Main Contribution Areas

- LPSup: calculus for Linear Temporal Logic (LTL)
- LS4: algorithm for LTL satisfiability based on SAT
- VCE: preprocessing method for LTL clause normal forms
- applied ideas to hardware verification
- further progressed to automated planning

Linear Temporal Logic

- propositional logic + temporal operators:
  - next: $\bigcirc$,
  - always: $\square$,
  - eventually: $\diamond$
  - $\ldots$

As a specification language

$$\square(sent \rightarrow \diamond delivered) \wedge \square(delivered \rightarrow \bigcirc read)$$

Why prove LTL theorems?

- debugging specifications
- synthesis: precondition to realizability

Linear Temporal Logic

- propositional logic + temporal operators:
  - next: $\bigcirc$,
  - always: $\Box$,
  - eventually: $\Diamond$
  - ...

As a specification language

$$\Box(\mathit{sent} \rightarrow \Diamond \mathit{delivered}) \wedge \Box(\mathit{delivered} \rightarrow \bigcirc \mathit{read})$$

Why prove LTL theorems?

- debugging specifications
- synthesis: precondition to realizability

Linear Temporal Logic

- propositional logic + temporal operators:
  - next: $\bigcirc$,
  - always: $\square$,
  - eventually: $\diamond$
  - $\ldots$

As a specification language

$$\square(\mathit{sent} \to \diamond \mathit{delivered}) \wedge \square(\mathit{delivered} \to \bigcirc \mathit{read})$$

Why prove LTL theorems?

- debugging specifications
- synthesis: precondition to realizability

LPSup: Labeled Superposition for LTL

- adapted superposition to deal with linear time
- new calculus LPSup
- inherits desired properties
  - ordering restrictions
  - completeness justifies abstract redundancy
  - backtrack-free model building

Main challenges

- appropriate clausal normal form
- keeping track of temporal dependencies
- detecting ultimately UNSAT instances

[Suda and Weidenbach, LPAR 2012]

LPSup: Labeled Superposition for LTL

- adapted superposition to deal with linear time
- new calculus LPSup
- inherits desired properties
  - ordering restrictions
  - completeness justifies abstract redundancy
  - backtrack-free model building

Main challenges

- appropriate clausal normal form
- keeping track of temporal dependencies
- detecting ultimately UNSAT instances

[Suda and Weidenbach, LPAR 2012]

LTL Clause Normal Forms

- SNF [Fisher 1991]
- TST: Initial clauses I, step clauses T, and goal clauses G

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

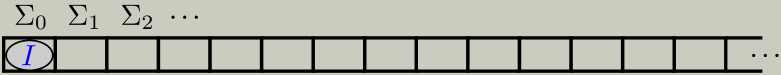
Semantics in a picture

LTL Clause Normal Forms

- SNF [Fisher 1991]
- TST: 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

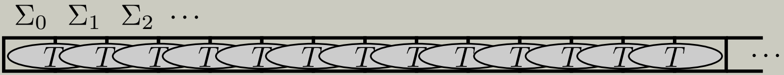


LTL Clause Normal Forms

- SNF [Fisher 1991]
- TST: 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

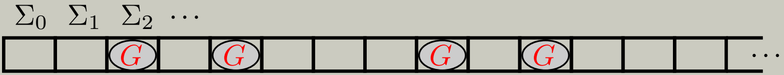


LTL Clause Normal Forms

- SNF [Fisher 1991]
- TST: 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



Idea of Labels

- cast to standard propositional satisfiability
  - infinitely many copies
  - infinitely many configurations
- finitely represent using labels
- uniformly lifted in labeled inferences

Labeled resolution inference

$$\mathcal{I}\frac{L_1 \,||\, C \vee a \qquad L_2 \,||\, D \vee \neg a}{(L_1 \sqcap L_2) \,||\, C \vee D}$$

- L_1 and L_2 *merged* to express intersection of the temporal contexts

Idea of Labels

- cast to standard propositional satisfiability
  - infinitely many copies
  - infinitely many configurations
- finitely represent using labels
- uniformly lifted in labeled inferences

Labeled resolution inference

$$\mathcal{I}\,\frac{L_1 \mid\mid C \vee a \qquad L_2 \mid\mid D \vee \neg a}{(L_1 \sqcap L_2) \mid\mid C \vee D}$$

- L_1 and L_2 *merged* to express intersection of the temporal contexts

To Make it Complete

- several kinds of empty clauses
- potentially infinite derivations

- special saturation strategy
- repetition detection and derivation replaying argument

"Structural" inference Leap

$$\mathcal{I}\frac{\{(b, u + i \cdot v) \,||\, C\}_{i \in \mathbb{N}} \text{ derivable from } N}{(b, u - v) \,||\, C}$$

where $u \geq v > 0$ are integers and C is an arbitrary standard clause

- Leap eliminates worlds that cannot reach themselves

To Make it Complete

- several kinds of empty clauses
- potentially infinite derivations

- special saturation strategy
- repetition detection and derivation replaying argument

"Structural" inference Leap

$$\mathcal{I}\,\frac{\{(b, u + i \cdot v) \,||\, C\}_{i \in \mathbb{N}} \text{ derivable from } N}{(b, u - v) \,||\, C}$$

where $u \geq v > 0$ are integers and C is an arbitrary standard clause

- Leap eliminates worlds that cannot reach themselves

SAT Solver Instead of Saturation

- connection between superposition and CDCL [Weidenbach]
- model-guidance idea:
  - build a partial model on the fly
  - derive clauses only to resolve conflicts during model construction

LS4: a new algorithm for LTL satisfiability based on SAT

- maintains connection to LPSup on macro-level
- efficient SAT solver as a black-box on micro-level
- one of the strongest LTL solvers

[Suda and Weidenbach, IJCAR 2012]

SAT Solver Instead of Saturation

- connection between superposition and CDCL [Weidenbach]
- model-guidance idea:
  - build a partial model on the fly
  - derive clauses only to resolve conflicts during model construction

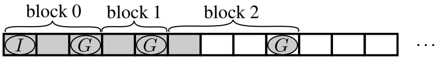
LS4: a new algorithm for LTL satisfiability based on SAT

- maintains connection to LPSup on macro-level
- efficient SAT solver as a black-box on micro-level
- one of the strongest LTL solvers

[Suda and Weidenbach, IJCAR 2012]

LS4 – Algorithm

- eager *forward* model construction



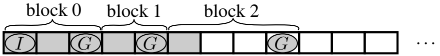
- model repetition check
- clauses learned *backward* when the “extension” fails
- clause layer repetition check

Used technology

- SAT solving under assumptions
- marking literals

LS4 – Algorithm

- eager *forward* model construction



- model repetition check
- clauses learned *backward* when the “extension” fails
- clause layer repetition check

Used technology

- SAT solving under assumptions
- marking literals

LS4 – Implementation

- approx 1k LOC of C++
- MiniSat 2.2 inside
- publicly available source

Success stories

- LTL backend in the TLA+ prover
- HWMCC'14 – liveness track
  - 5 unique solutions
- one of the best publicly available LTL provers
  - standard LTL benchmark suite [Schuppan and Darmawan, 2011]

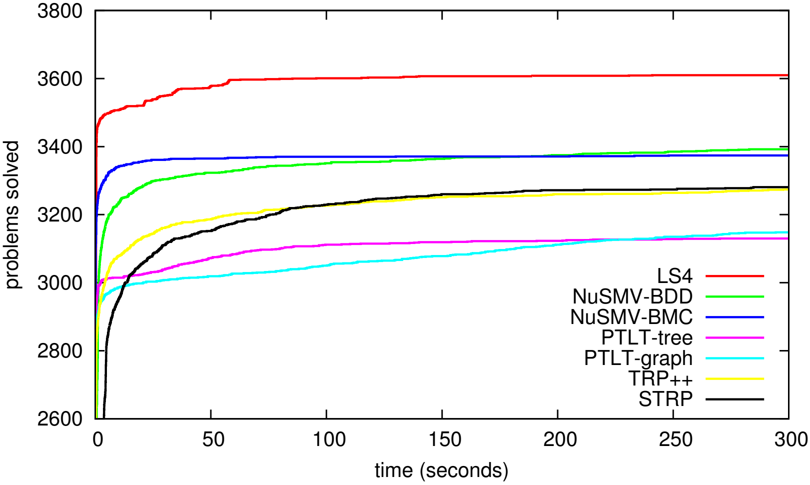
LS4 – Implementation

- approx 1k LOC of C++
- MiniSat 2.2 inside
- publicly available source

Success stories

- LTL backend in the TLA+ prover
- HWMCC'14 – liveness track
  - 5 unique solutions
- one of the best publicly available LTL provers
  - standard LTL benchmark suite [Schuppan and Darmawan, 2011]

Experimental Comparison



Variable and Clause Elimination

- useful preprocessing technique
  - simplify clausal input before solving
  - removes inefficiencies of a normal form transformation
- originally from SAT [Eén and Biere, 2005]

VCE: Variable and clause elimination for LTL

- adapted variable and clause elimination to LTL
- extend version of labeled clauses
- implementation prototype
  - shown practically effective

[Suda, MACIS 2013] ([Suda, MCS 2015])

Variable and Clause Elimination

- useful preprocessing technique
  - simplify clausal input before solving
  - removes inefficiencies of a normal form transformation
- originally from SAT [Eén and Biere, 2005]

VCE: Variable and clause elimination for LTL

- adapted variable and clause elimination to LTL
- extend version of labeled clauses
- implementation prototype
  - shown practically effective

[Suda, MACIS 2013] ([Suda, MCS 2015])

Variable Elimination Details

- clause distribution rule

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

Adapting to LTL

- labels from LPSup extended
- theorem: finitely many “exotic” clauses can be ignored
- some inherent limitations (due to expressiveness)

Variable Elimination Details

- clause distribution rule

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

Adapting to LTL

- labels from LPSup extended
- theorem: finitely many “exotic” clauses can be ignored
- some inherent limitations (due to expressiveness)

Variable Elimination Details

- clause distribution rule

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

Adapting to LTL

- labels from LPSup extended
- theorem: finitely many “exotic” clauses can be ignored
- some inherent limitations (due to expressiveness)

Experiment

Prototype implementation

- reuse MiniSat's simplification loop
- emulate labels by marking literals

- results on the standard LTL benchmark suite
  - eliminated 39% of the variables (7% original, 32% auxiliary)
  - eliminated 32% of clauses
  - both LS4 and trp++ solved more problems and faster on average

Further potential

- exploit the theory in full
- lift other preprocessing techniques
  - blocked clause elimination [Järvisalo et al., 2010]

Experiment

Prototype implementation

- reuse MiniSat's simplification loop
- emulate labels by marking literals

- results on the standard LTL benchmark suite
  - eliminated 39% of the variables (7% original, 32% auxiliary)
  - eliminated 32% of clauses
  - both LS4 and trp++ solved more problems and faster on average

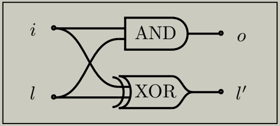
Further potential

- exploit the theory in full
- lift other preprocessing techniques
  - blocked clause elimination [Järvisalo et al., 2010]

Hardware Verification

- important part of standard industrial workflows

Example sequential circuit



$$o \leftarrow l \wedge i$$

$$l' \leftarrow l \oplus i$$

- temporal aspect from modeling registers

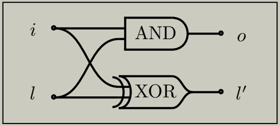
Verification of invariance and reachability

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

Hardware Verification

- important part of standard industrial workflows

Example sequential circuit



$o \leftarrow l \wedge i$

$l' \leftarrow l \oplus i$

- temporal aspect from modeling registers

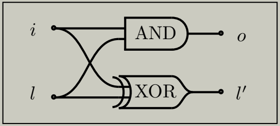
Verification of invariance and reachability

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

Hardware Verification

- important part of standard industrial workflows

Example sequential circuit



$$o \leftarrow l \wedge i$$

$$l' \leftarrow l \oplus i$$

- temporal aspect from modeling registers

Verification of invariance and reachability

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

Transfer Ideas to Hardware Verification

Reach

- new algorithm for verifying invariance
- LS4 specialized to reachability
- adapted to finite path semantics

Related work from hardware verification

- Bounded model checking [Biere et al., 1999]
  - Reach explores the same unrolling
- Interpolation-based model checking [McMillan, 2003]
  - clause layers in Reach are interpolants
- Property Directed Reachability [Bradley, 2011], [Eén et al., 2011]
  - where is the difference?

Transfer Ideas to Hardware Verification

Reach

- new algorithm for verifying invariance
- LS4 specialized to reachability
- adapted to finite path semantics

Related work from hardware verification

- Bounded model checking [Biere et al., 1999]
  - Reach explores the same unrolling
- Interpolation-based model checking [McMillan, 2003]
  - clause layers in Reach are interpolants
- Property Directed Reachability [Bradley, 2011], [Eén et al., 2011]
  - where is the difference?

From Reach to Property Directed Reachability

- small conceptual change
  - monotone layers
- three independent enhancements
  - obligation rescheduling
  - clause propagation
  - explicit (inductive) minimization

Extensive experimental evaluation

- each enhancement independently
- various criteria: search direction, problem status

Triggered clause pushing

- new technique for improving PDR's clause propagation phase
- especially useful in the multi-property setting

From Reach to Property Directed Reachability

- small conceptual change
  - monotone layers
- three independent enhancements
  - obligation rescheduling
  - clause propagation
  - explicit (inductive) minimization

Extensive experimental evaluation

- each enhancement independently
- various criteria: search direction, problem status

Triggered clause pushing

- new technique for improving PDR's clause propagation phase
- especially useful in the multi-property setting

From Reach to Property Directed Reachability

- small conceptual change
  - monotone layers
- three independent enhancements
  - obligation rescheduling
  - clause propagation
  - explicit (inductive) minimization

Extensive experimental evaluation

- each enhancement independently
- various criteria: search direction, problem status

Triggered clause pushing

- new technique for improving PDR's clause propagation phase
- especially useful in the multi-property setting

Automated Planning

- classical branch of artificial intelligence
- given a formal description of a world + set of available actions look for a sequence of actions that achieve a specified goal

Example

Operator $unstack(X, Y)$

pre : $clear(X), on(X, Y), arm\text{-}empty$

add : $holding(X), clear(Y)$

del : $clear(X), on(X, Y), arm\text{-}empty$

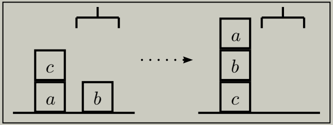
Industrial applications

- intelligent agents, autonomous robots, logistics, . . .

Automated Planning

- classical branch of artificial intelligence
- given a formal description of a world + set of available actions
  look for a sequence of actions that achieve a specified goal

Example



Operator $unstack(X, Y)$

$$\begin{aligned} \text{pre} &: clear(X), on(X,Y), arm\text{-}empty \\ \text{add} &: holding(X), clear(Y) \\ \text{del} &: clear(X), on(X,Y), arm\text{-}empty \end{aligned}$$

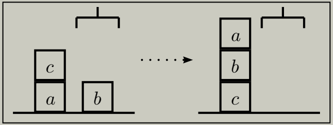
Industrial applications

- intelligent agents, autonomous robots, logistics, ...

Automated Planning

- classical branch of artificial intelligence
- given a formal description of a world + set of available actions look for a sequence of actions that achieve a specified goal

Example



$$\begin{aligned} &\text{Operator } \mathit{unstack}(X,Y) \\ \text{pre} &: \mathit{clear}(X), \mathit{on}(X,Y), \mathit{arm\text{-}empty} \\ \text{add} &: \mathit{holding}(X), \mathit{clear}(Y) \\ \text{del} &: \mathit{clear}(X), \mathit{on}(X,Y), \mathit{arm\text{-}empty} \end{aligned}$$

Industrial applications

- intelligent agents, autonomous robots, logistics, . . .

Property Directed Reachability for Automated Planning

1) via encodings from "Planning as SAT" [Kautz and Selman, 1992]

2) without a SAT solver
   - planning-specific procedure replaces the SAT calls
   - polynomial time upper bound on a single call
   - improvements beyond standard PDR

pdrPlan

- new planner based on 2)
- highly competitive for satisficing planning
- supports also: optimal planning, unsolvability detection

[Suda, JAIR 2014]

Conclusion

Summary

- Three resolution-based methods:
  - superposition (LPSup)
  - SAT solving (LS4)
  - clause distribution (VCE)
- Three application domains:
  - LTL proving
  - hardware verification
  - automated planning

Future work

- possible to extend beyond propositional logic
  - EPR, theories, . . .

Conclusion

Summary

- Three resolution-based methods:
  - superposition (LPSup)
  - SAT solving (LS4)
  - clause distribution (VCE)
- Three application domains:
  - LTL proving
  - hardware verification
  - automated planning

Future work

- possible to extend beyond propositional logic
  - EPR, theories, . . .