On Unit-Refutation Complete Formulae with Existentially Quantified Variables

Lucas Bordeaux[1] **Mikoláš Janota**[2]
Joao Marques-Silva[3] Pierre Marquis[4]

[1]Microsoft Research, Cambridge
[2]INESC-ID, Lisboa
[3]CASL, UCD & IST/INESC-ID
[4]CRIL-CNRS, U. d'Artois

CNF, Unit Propagation

- conjunctive normal form (CNF) is a popular language in solvers for its simple yet expressive structure
- unit propagation is an inference mechanism implemented virtually in all CNF-based solvers
- unit propagation can be computed polynomial time and moreover, efficient algorithms and data structures have been developed for it (*watch literals*)

CNF, Unit Propagation

- conjunctive normal form (CNF) is a popular language in solvers for its simple yet expressive structure
- unit propagation is an inference mechanism implemented virtually in all CNF-based solvers
- unit propagation can be computed polynomial time and moreover, efficient algorithms and data structures have been developed for it (*watch literals*)

Example of inference

Propagation

$$\boxed{\begin{array}{l} x \\ \bar{x} \vee y \\ \bar{y} \vee z \end{array}}$$

CNF, Unit Propagation

- conjunctive normal form (CNF) is a popular language in solvers for its simple yet expressive structure
- unit propagation is an inference mechanism implemented virtually in all CNF-based solvers
- unit propagation can be computed polynomial time and moreover, efficient algorithms and data structures have been developed for it (*watch literals*)

Example of inference

Propagation

$$\boxed{\begin{array}{l} x \\ \bar{x} \vee y \\ \bar{y} \vee z \end{array}} \qquad \vdash_u z$$

CNF, Unit Propagation

- conjunctive normal form (CNF) is a popular language in solvers for its simple yet expressive structure
- unit propagation is an inference mechanism implemented virtually in all CNF-based solvers
- unit propagation can be computed polynomial time and moreover, efficient algorithms and data structures have been developed for it (*watch literals*)

Example of inference

Propagation

$$\boxed{\begin{array}{l} x \\ \bar{x} \vee y \\ \bar{y} \vee z \end{array}} \qquad \vdash_u z$$

Refutation

$$\boxed{\begin{array}{c} \bar{x} \vee z \\ x \vee z \end{array}}$$

CNF, Unit Propagation

- conjunctive normal form (CNF) is a popular language in solvers for its simple yet expressive structure
- unit propagation is an inference mechanism implemented virtually in all CNF-based solvers
- unit propagation can be computed polynomial time and moreover, efficient algorithms and data structures have been developed for it (*watch literals*)

Example of inference

Propagation

$$\boxed{\begin{array}{l} x \\ \bar{x} \vee y \\ \bar{y} \vee z \end{array}} \quad \vdash_u z$$

Refutation

$$\boxed{\begin{array}{l} \bar{x} \vee z \\ x \vee z \\ \bar{z} \end{array}}$$

CNF, Unit Propagation

- conjunctive normal form (CNF) is a popular language in solvers for its simple yet expressive structure
- unit propagation is an inference mechanism implemented virtually in all CNF-based solvers
- unit propagation can be computed polynomial time and moreover, efficient algorithms and data structures have been developed for it (*watch literals*)

Example of inference

Propagation

$$\boxed{\begin{array}{l} x \\ \bar{x} \lor y \\ \bar{y} \lor z \end{array}} \qquad \vdash_u z$$

Refutation

$$\boxed{\begin{array}{l} \bar{x} \lor z \\ x \lor z \\ \bar{z} \end{array}} \qquad \vdash_u \bot$$

Unit Refutation Completeness

- for general CNF, unit propagation is not complete, i.e. "it does not let us infer all the facts"

Examples

$$\left.\begin{array}{|c|}\hline \bar{u} \vee w \\ u \vee \bar{w} \\ \bar{x} \vee \bar{u} \vee \bar{w} \\ \bar{x} \vee u \vee w \\ \\ \hline \end{array}\right. \quad \begin{array}{l} \models \bar{x} \\ \nvdash_u \bar{x} \end{array}$$

Unit Refutation Completeness

- for general CNF, unit propagation is not complete, i.e. “it does not let us infer all the facts”

Examples

$$\boxed{\begin{array}{l} \bar{u} \lor w \\ u \lor \bar{w} \\ \bar{x} \lor \bar{u} \lor \bar{w} \\ \bar{x} \lor u \lor w \end{array}} \quad \begin{array}{l} \models \bar{x} \\ \nvdash_u \bar{x} \end{array}$$

$$\boxed{\begin{array}{l} \bar{u} \lor w \\ u \lor \bar{w} \\ \bar{x} \lor \bar{u} \lor \bar{w} \\ \bar{x} \lor u \lor w \\ x \end{array}} \quad \nvdash_u \bot$$

Languages

Definition (URC-C)

a formula $\alpha \in \texttt{CNF}$ belongs to URC-C *iff* for every clause $\delta = l_1 \vee \cdots \vee l_k$

$$\textbf{if } \alpha \models \delta \textbf{ then } \alpha \wedge \bar{l}_1 \wedge ... \wedge \bar{l}_k \vdash_u \bot$$

Languages

Definition (URC-C)

a formula $\alpha \in$ CNF belongs to URC-C *iff* for every clause $\delta = l_1 \vee \cdots \vee l_k$

$$\textbf{if } \alpha \models \delta \textbf{ then } \alpha \wedge \bar{l}_1 \wedge \ldots \wedge \bar{l}_k \vdash_u \bot$$

Definition (Closures)

- if $\alpha_1 \vee \cdots \vee \alpha_n \in \mathcal{L}$ then $(\alpha_1 \vee \cdots \vee \alpha_n) \in \mathcal{L}[\vee]$

Languages

Definition (URC-C)

a formula $\alpha \in \texttt{CNF}$ belongs to URC-C *iff* for every clause $\delta = l_1 \vee \cdots \vee l_k$

$$\textbf{if } \alpha \models \delta \textbf{ then } \alpha \wedge \bar{l}_1 \wedge \ldots \wedge \bar{l}_k \vdash_u \bot$$

Definition (Closures)

- if $\alpha_1 \vee \cdots \vee \alpha_n \in \mathcal{L}$ then $(\alpha_1 \vee \cdots \vee \alpha_n) \in \mathcal{L}[\vee]$
- if $\alpha \in \mathcal{L}$ then $(\exists X.\ \alpha) \in \mathcal{L}[\exists]$

Languages

Definition (URC-C)

a formula $\alpha \in \texttt{CNF}$ belongs to URC-C *iff* for every clause $\delta = l_1 \vee \cdots \vee l_k$

$$\textbf{if } \alpha \models \delta \textbf{ then } \alpha \wedge \bar{l}_1 \wedge \ldots \wedge \bar{l}_k \vdash_u \bot$$

Definition (Closures)

- if $\alpha_1 \vee \cdots \vee \alpha_n \in \mathcal{L}$ then $(\alpha_1 \vee \cdots \vee \alpha_n) \in \mathcal{L}[\vee]$
- if $\alpha \in \mathcal{L}$ then $(\exists X.\ \alpha) \in \mathcal{L}[\exists]$
- $\mathcal{L}[\exists, \vee]$ enables both rules

Motivation

The Quest for The Perfect Knowledge Representation Structure

- inference should be fast (tractability)
- representation should not be too large (succinctness)

Motivation

The Quest for The Perfect Knowledge Representation Structure

- inference should be fast (tractability)
- representation should not be too large (succinctness)

If a formula is unit refutation complete ...

- which queries can be answered efficiently?
- how does the size of formulas correspond to other representations?

Knowledge Compilation Map

Succinctness ($\leq_s$, $\leq_p$)

- $\mathcal{L}_1 \leq_s \mathcal{L}_2$, if any formula in $\mathcal{L}_2$ can be equivalently expressed in polynomially sized formula from $\mathcal{L}_1$

Knowledge Compilation Map

Succinctness ($\leq_s$, $\leq_p$)

- $\mathcal{L}_1 \leq_s \mathcal{L}_2$, if any formula in $\mathcal{L}_2$ can be equivalently expressed in polynomially sized formula from $\mathcal{L}_1$
- $\mathcal{L}_1 \leq_p \mathcal{L}_2$, just as $\leq_p$ but we also have an polynomial algorithm for it

Knowledge Compilation Map

Succinctness ($\leq_s$, $\leq_p$)

- $\mathcal{L}_1 \leq_s \mathcal{L}_2$, if any formula in $\mathcal{L}_2$ can be equivalently expressed in polynomially sized formula from $\mathcal{L}_1$
- $\mathcal{L}_1 \leq_p \mathcal{L}_2$, just as $\leq_p$ but we also have an polynomial algorithm for it

Queries—can we decide in polynomial time...

consistency, clausal entailment, etc.

Knowledge Compilation Map

Succinctness ($\leq_s$, $\leq_p$)

- $\mathcal{L}_1 \leq_s \mathcal{L}_2$, if any formula in $\mathcal{L}_2$ can be equivalently expressed in polynomially sized formula from $\mathcal{L}_1$
- $\mathcal{L}_1 \leq_p \mathcal{L}_2$, just as $\leq_p$ but we also have an polynomial algorithm for it

Queries—can we decide in polynomial time...

consistency, clausal entailment, etc.

Transformations—can we construct in polynomial time...

Conditioning ($\alpha[x]$), disjunction ($\alpha_1 \vee \cdots \vee \alpha_n$), etc.

Trivia for URC-C

for $\alpha, \alpha_1, \alpha_2 \in \texttt{URC-C}$

- clausal entailment $\alpha \models \gamma$ can be decided in polynomial time
  - negate γ and run unit propagation

Trivia for URC-C

for $\alpha, \alpha_1, \alpha_2 \in \texttt{URC-C}$

- clausal entailment $\alpha \models \gamma$ can be decided in polynomial time
  - negate γ and run unit propagation
- consistency of α can be decided in polynomial time
  - check that the empty clause is an implicate of α

Trivia for URC-C

for $\alpha, \alpha_1, \alpha_2 \in \mathtt{URC\text{-}C}$

- clausal entailment $\alpha \models \gamma$ can be decided in polynomial time
  - negate γ and run unit propagation
- consistency of α can be decided in polynomial time
  - check that the empty clause is an implicate of α
- equivalence of α_1, α_2 decidable in polynomial time
  - check that each clause in α_1 is an implicate of α_2
  - check that each clause in α_2 is an implicate of α_1

Trivia for URC-C

for $\alpha, \alpha_1, \alpha_2 \in \texttt{URC-C}$

- clausal entailment $\alpha \models \gamma$ can be decided in polynomial time
  - negate γ and run unit propagation
- consistency of α can be decided in polynomial time
  - check that the empty clause is an implicate of α
- equivalence of α_1, α_2 decidable in polynomial time
  - check that each clause in α_1 is an implicate of α_2
  - check that each clause in α_2 is an implicate of α_1
- if $\beta \in \texttt{CNF}$ contains all of its prime implicates then $\beta \in \texttt{URC-C}$
  - if $\beta \models \gamma$, then there is a prime implicate $\gamma' \subseteq \gamma$ and immediately $\beta \wedge \neg\gamma' \vdash_u \bot$

Enabling Existential Variables

Motivation: Using fresh variables in CNF enables...

- polynomial Boolean logic representation (Tseitin)
- cardinality encodings
- other constraints, e.g. XOR($x_1, \ldots, x_n$)

Enabling Existential Variables

Motivation: Using fresh variables in CNF enables...

- polynomial Boolean logic representation (Tseitin)
- cardinality encodings
- other constraints, e.g. XOR($x_1, \ldots, x_n$)

Definition ($\exists$URC-C)

a formula $\exists X.\ \alpha \in \texttt{CNF}[\exists]$ belongs to $\exists$URC-C *iff* for every clause $\delta = l_1 \vee \cdots \vee l_k$

$$\textbf{if } \exists X.\ \alpha \models \delta \textbf{ then } \alpha \wedge \bar{l}_1 \wedge ... \wedge \bar{l}_k \vdash_u \bot$$

$\exists\texttt{URC-C} \sim_p \exists\texttt{URC-C}[\vee]$

$\alpha = (\exists X_1.\ \alpha_1) \vee \cdots \vee (\exists X_n.\ \alpha_n)$

$\exists\mathtt{URC\text{-}C} \sim_p \exists\mathtt{URC\text{-}C}[\vee]$

$\alpha = (\exists X_1.\ \alpha_1) \vee \cdots \vee (\exists X_n.\ \alpha_n)$

[prenex] $\alpha' = \exists X.\ \alpha_1 \vee \cdots \vee \alpha_n$

$\exists\texttt{URC-C} \sim_p \exists\texttt{URC-C}[\vee]$

$$\alpha = (\exists X_1.\ \alpha_1) \vee \cdots \vee (\exists X_n.\ \alpha_n)$$

[prenex] $\alpha' = \exists X.\ \alpha_1 \vee \cdots \vee \alpha_n$

[Tseitin] $\alpha'' = \exists X \tau_1 \ldots \tau_n.\quad \begin{array}{l} \bar{\tau}_1 \vee \alpha_1 \\ \ldots \\ \bar{\tau}_n \vee \alpha_n \\ \tau_1 \vee \cdots \vee \tau_n \end{array}$

$\exists\mathtt{URC\text{-}C} \sim_p \exists\mathtt{URC\text{-}C}[\vee]$

$$\alpha = (\exists X_1.\ \alpha_1) \vee \cdots \vee (\exists X_n.\ \alpha_n)$$

[prenex] $\alpha' = \exists X.\ \alpha_1 \vee \cdots \vee \alpha_n$

[Tseitin] $\alpha'' = \exists X \tau_1 \ldots \tau_n.\quad \begin{array}{l} \bar{\tau}_1 \vee \alpha_1 \\ \ldots \\ \bar{\tau}_n \vee \alpha_n \\ \tau_1 \vee \cdots \vee \tau_n \end{array}$

- α'' is not necessarily unit refutation complete!

$\exists\texttt{URC-C} \sim_p \exists\texttt{URC-C}[\vee]$

$$\alpha = (\exists X_1.\ \alpha_1) \vee \cdots \vee (\exists X_n.\ \alpha_n)$$

[prenex] $\alpha' = \exists X.\ \alpha_1 \vee \cdots \vee \alpha_n$

[Tseitin] $\alpha'' = \exists X \tau_1 \ldots \tau_n.\quad \begin{array}{l} \bar{\tau}_1 \vee \alpha_1 \\ \ldots \\ \bar{\tau}_n \vee \alpha_n \\ \tau_1 \vee \cdots \vee \tau_n \end{array}$

- α'' is not necessarily unit refutation complete!
- if $\alpha \models \gamma$, then $(\exists X_i.\ \alpha_i) \models \gamma$, for $i \in 1..n$

$\exists\texttt{URC-C} \sim_p \exists\texttt{URC-C}[\vee]$

$$\alpha = (\exists X_1.\ \alpha_1) \vee \cdots \vee (\exists X_n.\ \alpha_n)$$

[prenex] $\alpha' = \exists X.\ \alpha_1 \vee \cdots \vee \alpha_n$

[Tseitin] $\alpha'' = \exists X \tau_1 \ldots \tau_n.\quad \begin{array}{l} \bar{\tau}_1 \vee \alpha_1 \\ \ldots \\ \bar{\tau}_n \vee \alpha_n \\ \tau_1 \vee \cdots \vee \tau_n \end{array}$

- α'' is not necessarily unit refutation complete!

- if $\alpha \models \gamma$, then $(\exists X_i.\ \alpha_i) \models \gamma$, for $i \in 1..n$
- since $\alpha_i \in \exists\texttt{URC-C}$, then $\alpha'' \wedge \tau_i \wedge \neg\gamma \vdash_u \perp$, for $i \in 1..n$

$\exists\mathtt{URC\text{-}C} \sim_p \exists\mathtt{URC\text{-}C}[\vee]$

$$
\begin{array}{rl}
 & \alpha = (\exists X_1.\ \alpha_1) \vee \cdots \vee (\exists X_n.\ \alpha_n) \\
\text{[prenex]} & \alpha' = \exists X.\ \alpha_1 \vee \cdots \vee \alpha_n \\
\text{[Tseitin]} & \alpha'' = \exists X \tau_1 \ldots \tau_n.\quad \begin{array}{l} \bar{\tau}_1 \vee \alpha_1 \\ \ldots \\ \bar{\tau}_n \vee \alpha_n \\ \tau_1 \vee \cdots \vee \tau_n \end{array}
\end{array}
$$

- α'' is not necessarily unit refutation complete!

- if $\alpha \models \gamma$, then $(\exists X_i.\ \alpha_i) \models \gamma$, for $i \in 1..n$
- since $\alpha_i \in \exists\mathtt{URC\text{-}C}$, then $\alpha'' \wedge \tau_i \wedge \neg\gamma \vdash_u \bot$, for $i \in 1..n$
- add new variables that "simulate" unit propagation on the disjuncts and derive $\bot$ if all derive $\bot$

Queries Results

$\mathcal{L}$	**CO**	**VA**	**CE**	**IM**	**EQ**	**SE**	**CT**	**ME**	**MC**
$\exists$URC-C	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
URC-C[$\vee, \exists$]	$\surd$	$\circ$	$\surd$	$\circ$	$\circ$	$\circ$	$\circ$	$\surd$	$\surd$
URC-C	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\circ$	$\surd$	$\surd$

$\surd$ means “satisfies”

$\circ$ means “does not satisfy unless P=NP”

Succinctness Results

1. $\exists$URC-C $\leq_s$ URC-C[$\vee, \exists$] $<_s$ URC-C $<_s$ PI
2. URC-C $\not\leq_s^*$ CNF and CNF $\leq_s$ URC-C
3. $\exists$URC-C $\not\leq_s^*$ CNF and CNF $\not\leq_s$ $\exists$URC-C
4. URC-C $\not\leq_s$ DNF, URC-C $\not\leq_s$ SDNNF, and URC-C $\not\leq_s$ d-DNNF,
5. DNF $\not\leq_s$ URC-C, SDNNF $\not\leq_s$ URC-C, and FBDD $\not\leq_s$ URC-C
6. $\exists$URC-C $\leq_s$ DNNF
7. $\exists$URC-C $<_s$ DNF
8. $\exists$URC-C $<_s$ SDNNF
9. $\exists$URC-C $<_s^*$ d-DNNF

- $\mathcal{L}_1 \not\leq_s^* \mathcal{L}_2$ means that $\mathcal{L}_1$ is not at least as succinct as $\mathcal{L}_2$ unless PH collapses

Transformations Results

$\mathcal{L}$	**CD**	**FO**	**SFO**	$\wedge$ **C**	$\wedge$**BC**	$\vee$**C**	$\vee$**BC**	$\neg$**C**
$\exists$URC-C	$\surd$	$\surd$	$\surd$	$\circ$	$\circ$	$\surd$	$\surd$	$\circ$
URC-C[$\vee, \exists$]	$\surd$	$\surd$	$\surd$	$\circ$	$\circ$	$\surd$	$\surd$	$\circ$
URC-C	$\surd$	$\bullet$	?	$\circ$	$\circ$	$\bullet$	?	$\bullet$

- $\surd$ means "satisfies"
- $\bullet$ means "does not satisfy"
- $\circ$ means "does not satisfy unless P=NP"

Conclusions and Future Work

- we studied the unit refutation complete language URC-C and its existential extension ∃URC-C

Conclusions and Future Work

- we studied the unit refutation complete language URC-C and its existential extension ∃URC-C
- the languages have number of favorable KR properties

Conclusions and Future Work

- we studied the unit refutation complete language URC-C and its existential extension ∃URC-C
- the languages have number of favorable KR properties
- URC-C is powerful in answering queries, e.g. clausal entailment, equivalence

Conclusions and Future Work

- we studied the unit refutation complete language URC-C and its existential extension ∃URC-C
- the languages have number of favorable KR properties
- URC-C is powerful in answering queries, e.g. clausal entailment, equivalence
- ∃URC-C loses some ability to answer queries but is (strictly) more succinct than number of interesting languages

Conclusions and Future Work

- we studied the unit refutation complete language `URC-C` and its existential extension ∃`URC-C`
- the languages have number of favorable KR properties
- `URC-C` is powerful in answering queries, e.g. clausal entailment, equivalence
- ∃`URC-C` loses some ability to answer queries but is (strictly) more succinct than number of interesting languages
- in the future we are interested in practical algorithms for compilation into `URC-C`

Conclusions and Future Work

- we studied the unit refutation complete language `URC-C` and its existential extension ∃`URC-C`
- the languages have number of favorable KR properties
- `URC-C` is powerful in answering queries, e.g. clausal entailment, equivalence
- ∃`URC-C` loses some ability to answer queries but is (strictly) more succinct than number of interesting languages
- in the future we are interested in practical algorithms for compilation into `URC-C`
- how can existential variables be employed (∃`URC-C`)