On Unit-Refutation Complete Formulae with Existentially Quantified Variables

Lucas Bordeaux\textsuperscript{1}  \hspace{1em} Mikoláš Janota\textsuperscript{2}  
Joao Marques-Silva\textsuperscript{3}  \hspace{1em} Pierre Marquis\textsuperscript{4}

\textsuperscript{1}Microsoft Research, Cambridge  
\textsuperscript{2}INESC-ID, Lisboa  
\textsuperscript{3}CASL, UCD & IST/INESC-ID  
\textsuperscript{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

\[
\begin{align*}
\neg x \\
\neg x \lor y \\
\neg y \lor z
\end{align*}
\]
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

\[\begin{array}{c}
x \\
\bar{x} \lor y \\
\bar{y} \lor z \\
\hline
\end{array}\]

\[\vdash_{\mathcal{U}} z\]
**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**

<table>
<thead>
<tr>
<th>Propagation</th>
<th>Refutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$\neg x \lor z$</td>
</tr>
<tr>
<td>$\neg x \lor y$</td>
<td>$x \lor z$</td>
</tr>
<tr>
<td>$\neg y \lor z$</td>
<td></td>
</tr>
</tbody>
</table>

$\vdash_{u} z$
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**

| $x$ | $\overline{x} \lor y$ | $\overline{y} \lor z$ |

$\vdash_{u} z$

**Refutation**

| $\overline{x} \lor z$ | $x \lor z$ | $\overline{z}$ |
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

<table>
<thead>
<tr>
<th>Propagation</th>
<th>Refutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$\overline{x} \lor z$</td>
</tr>
<tr>
<td>$\overline{x} \lor y$</td>
<td>$x \lor z$</td>
</tr>
<tr>
<td>$\overline{y} \lor z$</td>
<td>$\overline{z}$</td>
</tr>
</tbody>
</table>

$\vdash_u z$  

$\vdash_u \bot$
Unit Refutation Completeness

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

**Examples**

\[
\begin{align*}
\overline{u} \lor w \\
u \lor \overline{w} \\
\overline{x} \lor \overline{u} \lor \overline{w} \\
\overline{x} \lor u \lor w
\end{align*}
\]

\[
\models \overline{x} \\
\not\models_{u} \overline{x}
\]
Unit Refutation Completeness

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

Examples

\[
\begin{align*}
\overline{u} \lor w \\
u \lor \overline{w} \\
\overline{x} \lor \overline{u} \lor \overline{w} \\
\overline{x} \lor u \lor w
\end{align*}
\]

\[
\models \overline{x}
\]

\[
\not\models \overline{u} \bot
\]

\[
\begin{align*}
\overline{u} \lor w \\
u \lor \overline{w} \\
\overline{x} \lor \overline{u} \lor \overline{w} \\
\overline{x} \lor u \lor w \\
x
\end{align*}
\]
Languages

Definition (URC–C)

A formula $\alpha \in \text{CNF}$ belongs to URC–C iff for every clause $\delta = l_1 \lor \cdots \lor l_k$

$$ \text{if } \alpha \models \delta \text{ then } \alpha \land \overline{l}_1 \land \cdots \land \overline{l}_k \vdash u \bot $$
Languages

Definition (URC–C)

A formula $\alpha \in \text{CNF}$ belongs to URC–C iff for every clause

$$\delta = l_1 \lor \cdots \lor l_k$$

$$\text{if } \alpha \models \delta \text{ then } \alpha \land \overline{l}_1 \land \cdots \land \overline{l}_k \not\models u$$

Definition (Closures)

- If $\alpha_1 \lor \cdots \lor \alpha_n \in \mathcal{L}$ then $(\alpha_1 \lor \cdots \lor \alpha_n) \in \mathcal{L}[\lor]$
Languages

Definition (URC–C)

A formula $\alpha \in \text{CNF}$ belongs to URC–C iff for every clause $\delta = l_1 \lor \cdots \lor l_k$

\[
\text{if } \alpha \models \delta \text{ then } \alpha \land \overline{l_1} \land \cdots \land \overline{l_k} \vdash \bot
\]

Definition (Closures)

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

Definition (URC-C)
a formula $\alpha \in \text{CNF}$ belongs to URC-C iff for every clause
$\delta = l_1 \lor \cdots \lor l_k$

$$\text{if } \alpha \models \delta \text{ then } \alpha \land \bar{l}_1 \land \cdots \land \bar{l}_k \nvdash u \bot$$

Definition (Closures)

- if $\alpha_1 \lor \cdots \lor \alpha_n \in \mathcal{L}$ then $(\alpha_1 \lor \cdots \lor \alpha_n) \in \mathcal{L}[\lor]$
- if $\alpha \in \mathcal{L}$ then $(\exists X. \alpha) \in \mathcal{L}[\exists]$
- $\mathcal{L}[\exists, \lor]$ 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 a polynomial algorithm for it
Knowledge Compilation Map

Succinctness ($\leq_s, \leq_p$)

- $L_1 \leq_s L_2$, if any formula in $L_2$ can be equivalently expressed in polynomially sized formula from $L_1$
- $L_1 \leq_p L_2$, just as $\leq_p$ but we also have a 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 \((\alpha_1 \lor \cdots \lor \alpha_n)\), etc.
Triva for URC–C

for \( \alpha, \alpha_1, \alpha_2 \in \text{URC–C} \)

- clausal entailment \( \alpha \models \gamma \) can be decided in polynomial time
  - negate \( \gamma \) and run unit propagation
Trivia for URC–C

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

- **clausal entailment** $\alpha \models \gamma$ can be decided in polynomial time
  - negate $\gamma$ and run unit propagation
- **consistency** of $\alpha$ can be decided in polynomial time
  - check that the empty clause is an implicate of $\alpha$
for $\alpha, \alpha_1, \alpha_2 \in \text{URC-C}$

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

- **consistency** of $\alpha$ can be decided in polynomial time
  - check that the empty clause is an implicate of $\alpha$

- **equivalence** of $\alpha_1, \alpha_2$ decidable in polynomial time
  - check that each clause in $\alpha_1$ is an implicate of $\alpha_2$
  - check that each clause in $\alpha_2$ is an implicate of $\alpha_1$
Trivia for URC–C

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

- **clausal entailment** $\alpha \models \gamma$ can be decided in polynomial time
  - negate $\gamma$ and run unit propagation
- **consistency** of $\alpha$ can be decided in polynomial time
  - check that the empty clause is an implicate of $\alpha$
- **equivalence** of $\alpha_1, \alpha_2$ decidable in polynomial time
  - check that each clause in $\alpha_1$ is an implicate of $\alpha_2$
  - check that each clause in $\alpha_2$ is an implicate of $\alpha_1$
- if $\beta \in \text{CNF}$ contains all of its prime implicants then $\beta \in \text{URC–C}$
  - if $\beta \models \gamma$, then there is a prime implicate $\gamma' \subseteq \gamma$ and immediately $\beta \wedge \neg \gamma' \not\vdash_u \bot$
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 (∃URC–C)

A formula $\exists X. \alpha \in \text{CNF}[\exists]$ belongs to $\exists\text{URC–C}$ iff for every clause $\delta = l_1 \lor \cdots \lor l_k$

$$\text{if } \exists X. \alpha \models \delta \text{ then } \alpha \land \bar{l}_1 \land \cdots \land \bar{l}_k \models u \perp$$
\( \exists \text{URC-C} \sim_p \exists \text{URC-C} [\lor] \)

\[ \alpha = (\exists X_1. \alpha_1) \lor \cdots \lor (\exists X_n. \alpha_n) \]
∃URC-C ≈_{p} ∃URC-C[\lor]

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

[prenex] \alpha' = \exists X. \alpha_1 \lor \cdots \lor \alpha_n
$$\exists \text{URC-C} \sim_p \exists \text{URC-C}[\lor]$$

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

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

[Tseitin] $$\alpha'' = \exists X \tau_1 \cdots \tau_n. \bar{\tau}_1 \lor \alpha_1$$
$$\cdots$$
$$\bar{\tau}_n \lor \alpha_n$$
$$\tau_1 \lor \cdots \lor \tau_n$$

α'' is not necessarily unit refutation complete!
\[ \exists \text{URC-C} \sim_\rho \exists \text{URC-C}[\lor] \]

\[ \alpha = (\exists X_1. \alpha_1) \lor \cdots \lor (\exists X_n. \alpha_n) \]

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

[Tseitin] \[ \alpha'' = \exists X \tau_1 \ldots \tau_n. \bar{\tau}_1 \lor \alpha_1 \]

\[ \cdots \]

\[ \bar{\tau}_n \lor \alpha_n \]

\[ \tau_1 \lor \cdots \lor \tau_n \]

\[ \alpha'' \text{ is not necessarily unit refutation complete!} \]
\[ \exists \text{URC-C} \sim_p \exists \text{URC-C}[\lor] \]

\[ \alpha = (\exists X_1. \alpha_1) \lor \cdots \lor (\exists X_n. \alpha_n) \]

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

[Tseitin] \[ \alpha'' = \exists X \tau_1 \ldots \tau_n. \bar{\tau}_1 \lor \alpha_1 \]

\[ \ldots \]

\[ \bar{\tau}_n \lor \alpha_n \]

\[ \tau_1 \lor \cdots \lor \tau_n \]

- \( \alpha'' \) is not necessarily unit refutation complete!

- If \( \alpha \models \gamma \), then \( (\exists X_i. \alpha_i) \models \gamma \), for \( i \in 1..n \)
∃URC-C \sim_p ∃URC-C[\lor]

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

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

[Tseitin] \alpha'' = \exists X \tau_1 \cdots \tau_n. \bar{\tau}_1 \lor \alpha_1

\cdots

\bar{\tau}_n \lor \alpha_n

\tau_1 \lor \cdots \lor \tau_n

\begin{itemize}
  \item \alpha'' \text{ is not necessarily unit refutation complete!}
  \item if \alpha \models \gamma, then (\exists X_i. \alpha_i) \models \gamma, for i \in 1..n
  \item since \alpha_i \in \exists\text{URC-C}, then \alpha'' \land \tau_i \land \neg \gamma \vdash_u \bot, for i \in 1..n
\end{itemize}
\[ \exists_{\text{URC-C}} \sim_p \exists_{\text{URC-C}}[\forall] \]

\[ \alpha = (\exists X_1. \alpha_1) \lor \cdots \lor (\exists X_n. \alpha_n) \]

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

[Tseitin] \[ \alpha'' = \exists X \tau_1 \ldots \tau_n. \bar{\tau}_1 \lor \alpha_1 \]
\[ \ldots \]
\[ \bar{\tau}_n \lor \alpha_n \]
\[ \tau_1 \lor \cdots \lor \tau_n \]

- \( \alpha'' \) is not necessarily unit refutation complete!

- if \( \alpha \models \gamma \), then \( (\exists X_i. \alpha_i) \models \gamma \), for \( i \in 1..n \)
- since \( \alpha_i \in \exists_{\text{URC-C}} \), then \( \alpha'' \land \tau_i \land \neg \gamma \vdash_u \bot \), for \( i \in 1..n \)
- add new variables that “simulate” unit propagation on the disjuncts and derive \( \bot \) if all derive \( \bot \)
## Queries Results

<table>
<thead>
<tr>
<th>$\mathcal{L}$</th>
<th>CO</th>
<th>VA</th>
<th>CE</th>
<th>IM</th>
<th>EQ</th>
<th>SE</th>
<th>CT</th>
<th>ME</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exists$URC-C</td>
<td>$\sqrt{}$</td>
<td>⬜</td>
<td>$\sqrt{}$</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
<td>$\sqrt{}$</td>
</tr>
<tr>
<td>URC-C[$\lor$, $\exists$]</td>
<td>$\sqrt{}$</td>
<td>⬜</td>
<td>$\sqrt{}$</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
<td>⬜</td>
<td>$\sqrt{}$</td>
</tr>
<tr>
<td>URC-C</td>
<td>$\sqrt{}$</td>
<td>$\sqrt{}$</td>
<td>$\sqrt{}$</td>
<td>$\sqrt{}$</td>
<td>$\sqrt{}$</td>
<td>$\sqrt{}$</td>
<td>⬜</td>
<td>$\sqrt{}$</td>
<td>$\sqrt{}$</td>
</tr>
</tbody>
</table>

$\sqrt{}$ means “satisfies”

○ means “does not satisfy unless P=NP”
Succinctness Results

1. \( \exists \) URC-C \( \leq_s \) URC-C[\( \lor, \exists \)] <\( s \) URC-C <\( s \) PI
2. URC-C \( \not\leq^* \) CNF and CNF \( \leq_s \) URC-C
3. \( \exists \) URC-C \( \not\leq^* \) 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 <\(^* \) d-DNNF

- \( \mathcal{L}_1 \not\leq^* \mathcal{L}_2 \) means that \( \mathcal{L}_1 \) is not at least as succinct as \( \mathcal{L}_2 \) unless PH collapses
### Transformations Results

<table>
<thead>
<tr>
<th>$\mathcal{L}$</th>
<th>CD</th>
<th>FO</th>
<th>SFO</th>
<th>$\land C$</th>
<th>$\land BC$</th>
<th>$\lor C$</th>
<th>$\lor BC$</th>
<th>$\neg C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exists URC-C$</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>○</td>
<td>○</td>
<td>√</td>
<td>√</td>
<td>○</td>
</tr>
<tr>
<td>$URC-C[\lor, \exists]$</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>○</td>
<td>○</td>
<td>√</td>
<td>√</td>
<td>○</td>
</tr>
<tr>
<td>$URC-C$</td>
<td>√</td>
<td>•</td>
<td>?</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>?</td>
<td>●</td>
</tr>
</tbody>
</table>

- √ means “satisfies”
- ● means “does not satisfy”
- ○ means “does not satisfy unless $P=NP$”
Conclusions and Future Work

- we studied the unit refutation complete language $\text{URC-C}$ and its existential extension $\exists \text{URC-C}$.
Conclusions and Future Work

- we studied the **unit refutation complete** language $\text{URC} - \text{C}$ and its existential extension $\exists \text{URC} - \text{C}$
- the languages have number of favorable KR properties
Conclusions and Future Work

- we studied the unit refutation complete language $\mathsf{URC-C}$ and its existential extension $\exists \mathsf{URC-C}$
- the languages have number of favorable KR properties
- $\mathsf{URC-C}$ is powerful in answering queries, e.g. clausal entailment, equivalence
Conclusions and Future Work

- we studied the unit refutation complete language $\text{URC-C}$ and its existential extension $\exists\text{URC-C}$
- the languages have number of favorable KR properties
- $\text{URC-C}$ is powerful in answering queries, e.g. clausal entailment, equivalence
- $\exists\text{URC-C}$ 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 $\text{URC-C}$ and its existential extension $\exists\text{URC-C}$
- the languages have number of favorable KR properties
- $\text{URC-C}$ is powerful in answering queries, e.g. clausal entailment, equivalence
- $\exists\text{URC-C}$ 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 $\text{URC-C}$
Conclusions and Future Work

- we studied the *unit refutation complete* language \(\text{URC-C}\) and its existential extension \(\exists\text{URC-C}\)
- the languages have number of favorable KR properties
- \(\text{URC-C}\) is powerful in answering queries, e.g. clausal entailment, equivalence
- \(\exists\text{URC-C}\) 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 \(\text{URC-C}\)
- how can existential variables be employed (\(\exists\text{URC-C}\))