# On Unit-Refutation Complete Formulae with Existentially Quantified Variables

Lucas Bordeaux<sup>1</sup> **Mikoláš Janota**<sup>2</sup> Joao Marques-Silva<sup>3</sup> Pierre Marquis<sup>4</sup>

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

- 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)

- 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



- 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)

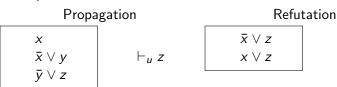
#### Example of inference

#### Propagation

$$\begin{array}{c|c} x \\ \bar{x} \lor y \\ \bar{y} \lor z \end{array} \vdash_{u} z$$

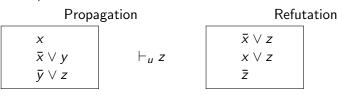
- conjunctive normal form (CNF) is a popular language in solvers for its simple yet expressive structure
- unit propagation is an inference mechanism implemented virtually in all CNF-based solvers
- unit propagation can be computed polynomial time and moreover, efficient algorithms and data structures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

#### Example of inference



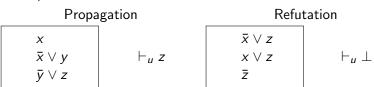
- conjunctive normal form (CNF) is a popular language in solvers for its simple yet expressive structure
- unit propagation is an inference mechanism implemented virtually in all CNF-based solvers
- unit propagation can be computed polynomial time and moreover, efficient algorithms and data structures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

#### Example of inference



- 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



## Unit Refutation Completeness

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

### **Examples**



## Unit Refutation Completeness

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

#### **Examples**





### Definition (URC-C)

a formula  $\alpha \in {\tt CNF}$  belongs to URC-C  $\it iff$  for every clause  $\delta = \it I_1 \lor \cdots \lor \it I_k$ 

if 
$$\alpha \models \delta$$
 then  $\alpha \wedge \overline{I}_1 \wedge ... \wedge \overline{I}_k \vdash_u \bot$ 

#### Definition (URC-C)

a formula  $\alpha \in \mathtt{CNF}$  belongs to URC-C  $\mathit{iff}$  for every clause  $\delta = \mathit{I}_1 \vee \cdots \vee \mathit{I}_k$ 

if 
$$\alpha \models \delta$$
 then  $\alpha \land \overline{I}_1 \land ... \land \overline{I}_k \vdash_u \bot$ 

#### Definition (Closures)

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

### Definition (URC-C)

a formula  $\alpha \in \mathtt{CNF}$  belongs to URC-C  $\mathit{iff}$  for every clause  $\delta = \mathit{I}_1 \vee \cdots \vee \mathit{I}_k$ 

if 
$$\alpha \models \delta$$
 then  $\alpha \land \overline{I}_1 \land ... \land \overline{I}_k \vdash_u \bot$ 

## 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  $(\exists X. \ \alpha) \in \mathcal{L}[\exists]$

### Definition (URC-C)

a formula  $\alpha \in \mathtt{CNF}$  belongs to URC-C  $\mathit{iff}$  for every clause  $\delta = \mathit{I}_1 \vee \cdots \vee \mathit{I}_k$ 

if 
$$\alpha \models \delta$$
 then  $\alpha \land \overline{I}_1 \land ... \land \overline{I}_k \vdash_u \bot$ 

## 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  $(\exists X. \ \alpha) \in \mathcal{L}[\exists]$
- $\mathcal{L}[\exists, \lor]$  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?

## 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  $\mathcal{L}_1$ 

## 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

## 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.

## 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.

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

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

- **clausal entailment**  $\alpha \models \gamma$  can be decided in polynomial time
  - ightharpoonup negate  $\gamma$  and run unit propagation
- lacktriangle consistency of lpha can be decided in polynomial time
  - lacktriangle check that the empty clause is an implicate of lpha
- **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  $\alpha_1$

- clausal entailment  $\alpha \models \gamma$  can be decided in polynomial time
  - ightharpoonup negate  $\gamma$  and run unit propagation
- lacktriangle consistency of lpha can be decided in polynomial time
  - lacktriangle check that the empty clause is an implicate of lpha
- **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  $\alpha_1$
- lacktriangleright if  $eta\in {\tt CNF}$  contains all of its prime implicates then  $eta\in {\tt URC-C}$ 
  - ▶ if  $\beta \models \gamma$ , then there is a prime implicate  $\gamma' \subseteq \gamma$  and immediately  $\beta \land \neg \gamma' \vdash_{u} \bot$

## **Enabling Existential Variables**

## Motivation: Using fresh variables in CNF enables...

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

## **Enabling Existential Variables**

## Motivation: Using fresh variables in CNF enables...

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

## Definition (∃URC-C)

a formula  $\exists X.\ \alpha \in \mathtt{CNF}[\exists]$  belongs to  $\exists \mathtt{URC-C}$  iff for every clause  $\delta = \mathit{I}_1 \lor \cdots \lor \mathit{I}_k$ 

if 
$$\exists X. \ \alpha \models \delta \text{ then } \alpha \land \overline{l}_1 \land ... \land \overline{l}_k \vdash_u \bot$$

$$\exists URC-C \sim_{p} \exists URC-C[\lor]$$

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

## $\exists URC-C \sim_{p} \exists URC-C[\lor]$

$$\alpha = (\exists X_1. \ \alpha_1) \lor \cdots \lor (\exists X_n. \ \alpha_n)$$
[prenex]  $\alpha' = \exists X. \ \alpha_1 \lor \cdots \lor \alpha_n$ 

## $\exists URC-C \sim_{p} \exists URC-C[\vee]$

$$\alpha = (\exists X_1. \ \alpha_1) \lor \cdots \lor (\exists X_n. \ \alpha_n)$$
[prenex]  $\alpha' = \exists X. \ \alpha_1 \lor \cdots \lor \alpha_n$ 
[Tseitin]  $\alpha'' = \exists X \tau_1 \dots \tau_n. \ \ \bar{\tau}_1 \lor \alpha_1$ 

$$\dots$$

$$\bar{\tau}_n \lor \alpha_n$$

$$\tau_1 \lor \cdots \lor \tau_n$$

## $\exists \mathtt{URC-C} \sim_{p} \exists \mathtt{URC-C}[\lor]$

$$\alpha = (\exists X_1. \ \alpha_1) \lor \cdots \lor (\exists X_n. \ \alpha_n)$$
[prenex]  $\alpha' = \exists X. \ \alpha_1 \lor \cdots \lor \alpha_n$ 
[Tseitin]  $\alpha'' = \exists X \tau_1 \dots \tau_n. \ \ \bar{\tau}_1 \lor \alpha_1$ 

$$\dots$$

$$\bar{\tau}_n \lor \alpha_n$$

$$\tau_1 \lor \cdots \lor \tau_n$$

 $\bullet$   $\alpha''$  is not necessarily unit refutation complete!

## $\exists \mathsf{URC} - \mathsf{C} \sim_{p} \exists \mathsf{URC} - \mathsf{C}[\vee]$

$$\alpha = (\exists X_1. \ \alpha_1) \lor \cdots \lor (\exists X_n. \ \alpha_n)$$
[prenex]  $\alpha' = \exists X. \ \alpha_1 \lor \cdots \lor \alpha_n$ 
[Tseitin]  $\alpha'' = \exists X \tau_1 \dots \tau_n. \ \ \bar{\tau}_1 \lor \alpha_1$ 

$$\dots$$

$$\bar{\tau}_n \lor \alpha_n$$

$$\tau_1 \lor \cdots \lor \tau_n$$

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

## $\exists URC-C \sim_{p} \exists URC-C[\vee]$

$$\alpha = (\exists X_1. \ \alpha_1) \lor \cdots \lor (\exists X_n. \ \alpha_n)$$
[prenex]  $\alpha' = \exists X. \ \alpha_1 \lor \cdots \lor \alpha_n$ 
[Tseitin]  $\alpha'' = \exists X \tau_1 \dots \tau_n. \ \ \bar{\tau}_1 \lor \alpha_1$ 

$$\dots$$

$$\bar{\tau}_n \lor \alpha_n$$

$$\tau_1 \lor \cdots \lor \tau_n$$

- $\bullet$   $\alpha''$  is not necessarily unit refutation complete!
- if  $\alpha \models \gamma$ , then  $(\exists X_i. \ \alpha_i) \models \gamma$ , for  $i \in 1..n$
- since  $\alpha_i \in \exists URC$ -C, then  $\alpha'' \wedge \tau_i \wedge \neg \gamma \vdash_u \bot$ , for  $i \in 1..n$

## $\exists \mathtt{URC-C} \sim_{p} \exists \mathtt{URC-C}[\lor]$

$$\alpha = (\exists X_1. \ \alpha_1) \lor \cdots \lor (\exists X_n. \ \alpha_n)$$
[prenex]  $\alpha' = \exists X. \ \alpha_1 \lor \cdots \lor \alpha_n$ 
[Tseitin]  $\alpha'' = \exists X \tau_1 \ldots \tau_n. \ \ \bar{\tau}_1 \lor \alpha_1$ 

$$\vdots \\ \bar{\tau}_n \lor \alpha_n$$

$$\tau_1 \lor \cdots \lor \tau_n$$

- $\bullet$   $\alpha''$  is not necessarily unit refutation complete!
- if  $\alpha \models \gamma$ , then  $(\exists X_i. \alpha_i) \models \gamma$ , for  $i \in 1..n$
- since  $\alpha_i \in \exists URC$ -C, then  $\alpha'' \wedge \tau_i \wedge \neg \gamma \vdash_{\mu} \bot$ , for  $i \in 1...n$
- lacksquare add new variables that "simulate" unit propagation on the disjuncts and derive ot if all derive ot

## Queries Results

$\mathcal{L}$	CO	VA	CE	IM	EQ	SE	СТ	ME	МС
∃URC-C		0		0	0	0	0		$\sqrt{}$
$\overline{\mathtt{URC-C}[\vee,\exists]}$		0		0	0	0	0		
URC-C							0		

- √ means "satisfies"
- o 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  $\leq_s^*$  CNF and CNF  $\leq_s$  URC-C
- 3.  $\exists URC-C \nleq_s^* CNF \text{ and } CNF \nleq_s \exists URC-C$
- 4. URC-C  $\not\leq_s$  DNF, URC-C  $\not\leq_s$  SDNNF, and URC-C  $\not\leq_s$  d-DNNF,
- 5. DNF ≤ URC-C, SDNNF ≤ URC-C, and FBDD ≤ URC-C
- 6.  $\exists URC-C <_s DNNF$
- 7.  $\exists URC-C <_{s} DNF$
- 8.  $\exists URC-C <_s SDNNF$
- 9.  $\exists URC-C <_s^* d-DNNF$ 
  - $\mathcal{L}_1 \nleq_s^* \mathcal{L}_2$  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	∧ <b>C</b>	∧BC	∨C	∨BC	¬C
∃URC-C				0	0			0
$URC-C[\lor,\exists]$				0	0			0
URC-C		•	?	0	0	•	?	•

- √ means "satisfies"
- means "does not satisfy"
- means "does not satisfy unless P=NP"

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

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

- 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

- 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

- 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

- 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)