A unified framework for Certificate and Compilation for QBF

Igor Stéphan & Benoit Da Mota

ICLA’09 – Chennai – India.
In few words
A unified framework for certificate and compilation for Quantified Boolean Formulae.
In few words

- A unified framework for certificate and compilation for Quantified Boolean Formulae.
- We provide in our unified framework:
A unified framework for certificate and compilation for Quantified Boolean Formulae.

We provide in our unified framework:

- a search-based algorithm to compute a certificate for the validity of a Quantified Boolean Formula and
In few words

- A unified framework for certificate and compilation for Quantified Boolean Formulae.
- We provide in our unified framework:
  - a search-based algorithm to compute a certificate for the validity of a Quantified Boolean Formula and
  - a search-based algorithm to compile a valid Quantified Boolean Formula.
What is a Quantified Boolean Formula or QBF?
What is a Quantified Boolean Formula or QBF?

- a propositional symbol or $\top$ or $\bot$;
Quantified Boolean Formula

What is a Quantified Boolean Formula or QBF?

- a propositional symbol or $\top$ or $\bot$;
- $\neg\phi$ with $\phi$ a QBF;
What is a Quantified Boolean Formula or QBF?

- a propositional symbol or $\top$ or $\bot$;
- $\neg \phi$ with $\phi$ a QBF;
- $(\phi_1 \circ \phi_2)$ with $\circ \in \{\land, \lor, \to, \leftrightarrow\}$ and $\phi_1, \phi_2$ some QBF;
Quantified Boolean Formula

What is a Quantified Boolean Formula or QBF?

- a propositional symbol or $\top$ or $\bot$;
- $\neg \phi$ with $\phi$ a QBF;
- $(\phi_1 \circ \phi_2)$ with $\circ \in \{\land, \lor, \to, \leftrightarrow\}$ and $\phi_1, \phi_2$ some QBF;
- $(\exists x \, \phi)$ or $(\forall x \, \phi)$ with $x$ a propositional symbol and $\phi$ a QBF.
Quantified Boolean Formula

What is a Quantified Boolean Formula or QBF?

- a propositional symbol or $\top$ or $\bot$;
- $\neg \phi$ with $\phi$ a QBF;
- $(\phi_1 \circ \phi_2)$ with $\circ \in \{\land, \lor, \rightarrow, \leftrightarrow\}$ and $\phi_1, \phi_2$ some QBF;
- $(\exists x \ \phi)$ or $(\forall x \ \phi)$ with $x$ a propositional symbol and $\phi$ a QBF.

$$\forall a \exists b \forall c((c \lor b) \land (b \rightarrow (\exists d ((c \rightarrow d) \land (c \lor (a \leftrightarrow \neg d)))))$$
Quantified Boolean Formula

What is a Quantified Boolean Formula or QBF?

- a propositional symbol or $\top$ or $\bot$;
- $\neg\phi$ with $\phi$ a QBF;
- $(\phi_1 \circ \phi_2)$ with $\circ \in \{\land, \lor, \rightarrow, \leftrightarrow\}$ and $\phi_1, \phi_2$ some QBF;
- $(\exists x \ \phi)$ or $(\forall x \ \phi)$ with $x$ a propositional symbol and $\phi$ a QBF.

$$\forall a \exists b \forall c ((c \lor b) \land (b \rightarrow (\exists d ((c \rightarrow d) \land (c \lor (a \leftrightarrow \neg d)))))$$

Some restrictions
Quantified Boolean Formula

What is a Quantified Boolean Formula or QBF?

- a propositional symbol or $\top$ or $\bot$;
- $\neg\phi$ with $\phi$ a QBF;
- $(\phi_1 \circ \phi_2)$ with $\circ \in \{\land, \lor, \to, \leftrightarrow\}$ and $\phi_1, \phi_2$ some QBF;
- $(\exists x \phi)$ or $(\forall x \phi)$ with $x$ a propositional symbol and $\phi$ a QBF.

$$\forall a \exists b \forall c ((c \lor b) \land (b \to (\exists d ((c \to d) \land (c \lor (a \leftrightarrow \neg d)))))$$

Some restrictions

- closed: every propositional symbol is under the scope of a quantifier.
Quantified Boolean Formula

What is a Quantified Boolean Formula or QBF?

- a propositional symbol or \( \top \) or \( \bot \);
- \( \neg \phi \) with \( \phi \) a QBF;
- \((\phi_1 \circ \phi_2)\) with \( \circ \in \{\land, \lor, \to, \leftrightarrow\}\) and \( \phi_1, \phi_2 \) some QBF;
- \((\exists x \phi)\) or \((\forall x \phi)\) with \( x \) a propositional symbol and \( \phi \) a QBF.

\[
\forall a \exists b \forall c((c \lor b) \land (b \to (\exists d ((c \to d) \land (c \lor (a \to \neg d)))))
\]

Some restrictions

- closed: every propositional symbol is under the scope of a quantifier.
- prenex: \( Q M \) with \( Q \) the binder (a string of quantifiers and propositional symbols) and \( M \) the matrix (a Boolean formula).
Semantics and validity of QBF

Semantics of QBF
Semantics and validity of QBF

Semantics of QBF

- Semantics of Boolean connectors is defined in standard way.
Semantics and validity of QBF

### Semantics of QBF

- Semantics of Boolean connectors is defined in standard way.
- \((\exists x \phi) = ([x \leftarrow \top](\phi) \lor [x \leftarrow \bot](\phi))\)
Semantics of QBF

- Semantics of Boolean connectors is defined in standard way.
- $(\exists x \phi) = ([x \leftarrow T](\phi) \lor [x \leftarrow \bot](\phi))$
- $(\forall x \phi) = ([x \leftarrow T](\phi) \land [x \leftarrow \bot](\phi))$
## Semantics and validity of QBF

### Semantics of QBF
- Semantics of Boolean connectors is defined in standard way.
- \((\exists x \, \phi) = ([x \leftarrow T](\phi)) \lor [x \leftarrow \bot](\phi))\)
- \((\forall x \, \phi) = ([x \leftarrow T](\phi)) \land [x \leftarrow \bot](\phi))\)

### Validity of QBF
- A quantified Boolean formula \(\phi\) is valid if \(\phi \equiv T\).
Semantics and validity of QBF

Semantics of QBF

- Semantics of Boolean connectors is defined in standard way.
- \((\exists x \phi) = ([x \leftarrow \top](\phi) \lor [x \leftarrow \bot](\phi))\)
- \((\forall x \phi) = ([x \leftarrow \top](\phi) \land [x \leftarrow \bot](\phi))\)

Validity of QBF

- A quantified Boolean formula \(\phi\) is valid if \(\phi \equiv \top\).
- Decision problem for validity of QBF is PSPACE-complete.
Semantics of QBF

- Semantics of Boolean connectors is defined in standard way.
- $$(\exists x \phi) = ([x \leftarrow T](\phi) \lor [x \leftarrow \bot](\phi))$$
- $$(\forall x \phi) = ([x \leftarrow T](\phi) \land [x \leftarrow \bot](\phi))$$

Validity of QBF

- A quantified Boolean formula $\phi$ is valid if $\phi \equiv T$.
- Decision problem for validity of QBF is PSPACE-complete.

- $$\forall c \exists a \exists b((b \lor a) \leftrightarrow c)$$ is valid.
Semantics and validity of QBF

Semantics of QBF

- Semantics of Boolean connectors is defined in standard way.
- $(\exists x \phi) = ([x \leftarrow \top](\phi) \lor [x \leftarrow \bot](\phi))$
- $(\forall x \phi) = ([x \leftarrow \top](\phi) \land [x \leftarrow \bot](\phi))$

Validity of QBF

- A quantified Boolean formula $\phi$ is valid if $\phi \equiv \top$.
- Decision problem for validity of QBF is PSPACE-complete.

- $\forall c \exists a \exists b((b \lor a) \leftrightarrow c)$ is valid.
- $\exists a \exists b \forall c((b \lor a) \leftrightarrow c)$ is not valid.
Models for QBF
Let $QF$ be a prenex QBF and $y^1, \ldots, y^p$ its existentially quantified propositional symbols.
Models for QBF

- Let \( QF \) be a prenex QBF and \( y^1, \ldots, y^p \) its existentially quantified propositional symbols.
- Let \( \hat{y}^i_{x_1 \ldots x_n} \) be a Boolean formula such that \( \{x_1 \ldots x_n\} \) are the universally quantified propositional symbols which precede \( y^i \).
Models for QBF

- Let $QF$ be a prenex QBF and $y^1, \ldots, y^p$ its existentially quantified propositional symbols.
- Let $\hat{y}^i_{x_1 \ldots x_n}$ be a Boolean formula such that $\{x_1 \ldots x_n\}$ are the universally quantified propositional symbols which precede $y^i$.

Model for valid QBF

$\hat{y}^1; \ldots; \hat{y}^p$ is a model for $QF$ if $[y^1 \leftarrow \hat{y}^1] \ldots [y^p \leftarrow \hat{y}^p](F)$ is a tautology.
Let $QF$ be a prenex QBF and $y^1, \ldots, y^p$ its existentially quantified propositional symbols.

Let $\hat{y}^i_{x_1 \ldots x_n}$ be a Boolean formula such that $\{x_1 \ldots x_n\}$ are the universally quantified propositional symbols which precede $y^i$.

**Model for valid QBF**

$\hat{y}^1; \ldots; \hat{y}^p$ is a model for $QF$ if $[y^1 \leftarrow \hat{y}^1] \ldots [y^p \leftarrow \hat{y}^p](F)$ is a tautology.

**Model-equivalence**

$QF \equiv QF'$ if $QF$ and $QF'$ have same models.
∀c∃a∃b((b ∨ a) ↔ c)
∀c∃a∃b((b ∨ a) ↔ c)
∀c∃a∃b((b ∨ a) ↔ c)
\[\forall c \exists a \exists b ((b \lor a) \leftrightarrow c)\]
∀c ∃a ∃b((b ∨ a) ↔ c)
\( \forall c \exists a \exists b ((b \lor a) \leftrightarrow c) \)
∀c∃a∃b((b∨a)↔c)
∀c∃a∃b((b ∨ a) ⇔ c)
∀c∃a∃b((b ∨ a) ↔ c)
∀c∃a∃b((b ∨ a) ↔ c)
Everything is not so simple (first part)

Soundness of QBF solvers
Everything is not so simple (first part)

Soundness of QBF solvers

- Most of the QBF solvers are decision procedures
Everything is not so simple (first part)

Soundness of QBF solvers

- Most of the QBF solvers are decision procedures
- How to check the result of a computation?
Everything is not so simple (first part)

<table>
<thead>
<tr>
<th>Soundness of QBF solvers</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Most of the QBF solvers are decision procedures</td>
</tr>
<tr>
<td>- How to check the result of a computation?</td>
</tr>
</tbody>
</table>

| A solution: sat-certificate |
Soundness of QBF solvers

- Most of the QBF solvers are decision procedures
- How to check the result of a computation?

A solution: *sat*-certificate

- A certificate for a valid QBF is any piece of information that provides self-supporting evidence of validity for that QBF.
Everything is not so simple (first part)

**Soundness of QBF solvers**
- Most of the QBF solvers are decision procedures
- How to check the result of a computation?

**A solution : **sat-certificate
- A certificate for a valid QBF is any piece of information that provides self-supporting evidence of validity for that QBF.
- A **sat**-certificate is a representation of a sequence of pairs of Boolean functions for a QBF that certifies its validity.
Everything is not so simple (first part)

Soundness of QBF solvers

- Most of the QBF solvers are decision procedures
- How to check the result of a computation?

A solution: \textit{sat}-certificate

- A certificate for a valid QBF is any piece of information that provides self-supporting evidence of validity for that QBF.
- A \textit{sat}-certificate is a representation of a sequence of pairs of Boolean functions for a QBF that certifies its validity.
- In Benedetti: extracted a posteriori from a trace of a skolemization-based solver.
Our first contribution
Our first contribution

• Extracting $\text{sat}$-certificates during the decision process
Our first contribution

- Extracting **sat**-certificates during the decision process
- ...for search-based algorithms.
Our first contribution

- Extracting \textit{sat}-certificates during the decision process
- \ldots for search-based algorithms.
- An operator to build from the \textit{sat}-certificates for $Q[x \leftarrow \top](F)$ and $Q[x \leftarrow \bot](F)$ a \textit{sat}-certificate for $\exists x QF$ or $\forall x QF$. 
Our first contribution

- Extracting **sat**-certificates during the decision process
- ... for search-based algorithms.
- An operator to build from the **sat**-certificates for $Q[x \leftarrow \top](F)$ and $Q[x \leftarrow \bot](F)$ a **sat**-certificate for $\exists x QF$ or $\forall x QF$.
- A complete and sound search-based algorithm that extracts **sat**-certificates during the decision process.
QBF as a finite two-player game
QBF as a finite two-player game

- Every finite two-player game can be modeled by a QBF.
## QBF as a finite two-player game

- Every finite two-player game can be modeled by a QBF.
- Existential and universal players.
Everything is not so simple (second part)

QBF as a finite two-player game

- Every finite two-player game can be modeled by a QBF.
- Existential and universal players.
- If the QBF is valid, the $\exists$-player has always a way to win.
Everything is not so simple (second part)

QBF as a finite two-player game

- Every finite two-player game can be modeled by a QBF.
- Existential and universal players.
- If the QBF is valid, the $\exists$-player has always a way to win.
- A model explicits one $\exists$-move to win for each $\forall$-move.
Everything is not so simple (second part)

QBF as a finite two-player game

- Every finite two-player game can be modeled by a QBF.
- Existential and universal players.
- If the QBF is valid, the $\exists$-player has always a way to win.
- A model explicits one $\exists$-move to win for each $\forall$-move.

Next move choice problem
Everything is not so simple (second part)

QBF as a finite two-player game
- Every finite two-player game can be modeled by a QBF.
- Existential and universal players.
- If the QBF is valid, the $\exists$-player has always a way to win.
- A model explicits one $\exists$-move to win for each $\forall$-move.

Next move choice problem
- Instance: Let
  - $q_1x_1 \ldots q_nx_n M$ be a QBF
  - $[x_1 \leftarrow C_1] \ldots [x_i \leftarrow C_i]$ a substitution obtained from a model for $q_1x_1 \ldots q_nx_n M$ with $C_1, \ldots, C_i \in \{\top, \bot\}$
  - $q_i = \exists$
Everything is not so simple (second part)

**QBF as a finite two-player game**

- Every finite two-player game can be modeled by a QBF.
- Existential and universal players.
- If the QBF is valid, the $\exists$-player has always a way to win.
- A model explicit one $\exists$-move to win for each $\forall$-move.

---

**Next move choice problem**

- **Instance**: Let
  - $q_1x_1 \ldots q_nx_nM$ be a QBF
  - $[x_1 \leftarrow C_1] \ldots [x_i \leftarrow C_i]$ a substitution obtained from a model for $q_1x_1 \ldots q_nx_nM$ with $C_1, \ldots, C_i \in \{\top, \bot\}$
  - $q_i = \exists$

- **Query**: Does there exist a model for $q_{i+1} \ldots q_nx_n[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \overline{C_i}(M)$.
\[ F = \forall a \exists b \forall c \exists d Q \phi \]
$F = \forall a \exists b \forall c \exists d Q \phi$

- The sequence $\hat{b}_a = \top; \hat{d}_{ac} = (a \rightarrow c); \ldots$ is a model of $F$. 
\[ F = \forall a \exists b \forall c \exists d Q \phi \]

- The sequence \( \hat{b}_a = \top; \hat{d}_{ac} = (a \rightarrow c) \); \ldots is a model of \( F \).
- First move for the \( \forall \)-player is (arbitrarily) \( \bot \) for \( a \).

\[
\begin{array}{cc}
\top & \wedge \\
\bot & \\
b \quad b\?
\end{array}
\]
\[ F = \forall a \exists b \forall c \exists d Q \phi \]

- The sequence \( \hat{b}_a = \top; \hat{d}_{ac} = (a \rightarrow c); \ldots \) is a model of \( F \).
- First move for the \( \forall \)-player is (arbitrarily) \( \bot \) for \( a \).

\[
\begin{array}{ccc}
\top & \wedge & \bot \\
\\downarrow & & \\
\ & b & \ \ ?
\end{array}
\]

- Since \( \hat{b}_a = \top \), the first move for \( \exists \)-player is \( \top \) for \( b \).

\[
\begin{array}{ccc}
\top & \wedge & \bot \\
\ & b & \ \ \\
\ & \ & \ \\
\ & \ & \ \\
\ & \ & c \ ? & c
\end{array}
\]
• Second move for the $\forall$-player is (arbitrarily) $\top$ for $c$. 
\begin{itemize}
  \item Second move for the $\forall$-player is (arbitrarily) $\top$ for $c$.
  \item Since $\hat{d}_{ac} = (a \rightarrow c)$, the second move computed from the model for $\exists$-player is $\top$ for $d$.
\end{itemize}
• Second move for the $\forall$-player is (arbitrarily) $\top$ for $c$.

• Since $\hat{d}_{ac} = (a \rightarrow c)$, the second move computed from the model for $\exists$-player is $\top$ for $d$.

• Can the $\exists$-player move to $\bot$ and still be sure to win?
- Second move for the $\forall$-player is (arbitrarily) $\top$ for $c$.

- Since $\hat{d}_{ac} = (a \rightarrow c)$, the second move computed from the model for $\exists$-player is $\top$ for $d$.
- Can the $\exists$-player move to $\bot$ and still be sure to win?
- Is $Q[a \leftarrow \bot][b \leftarrow \top][c \leftarrow \top][d \leftarrow \bot](\phi)$ valid?
Second move for the $\forall$-player is (arbitrarily) $\top$ for $c$.

Since $\hat{d}_{ac} = (a \rightarrow c)$, the second move computed from the model for $\exists$-player is $\top$ for $d$.

Can the $\exists$-player move to $\bot$ and still be sure to win?

Is $Q[a \leftarrow \bot][b \leftarrow \top][c \leftarrow \top][d \leftarrow \bot](\phi)$ valid?

Is it always a PSPACE-problem?
Our second contribution
Our second contribution

- A compilation framework for QBF.
Our second contribution

- A compilation framework for QBF.
- An extension for search-based algorithms.
Our second contribution

- A compilation framework for QBF.
- An extension for search-based algorithms.
- An operator to build a compiled version of $\exists x QF$ or $\forall x QF$ from the compiled versions of $Q[x \leftarrow \top](F)$ and $Q[x \leftarrow \bot](F)$. 
Our second contribution

- A compilation framework for QBF.
- An extension for search-based algorithms.
- An operator to build a compiled version of $\exists x QF$ or $\forall x QF$ from the compiled versions of $Q[x \leftarrow \top](F)$ and $Q[x \leftarrow \bot](F)$.
- The next move problem is no more PSPACE-complete for the compiled QBF.
A unified framework for Certificate and Compilation for QBF
Literal base

A literal base is a pair \((Q, G)\) constituted
Literal base

A literal base is a pair \((Q, G)\) constituted

- either of \(Q = \epsilon\) and \(G = \top\), or \(G = \bot\),
## Literal base

A literal base is a pair $(Q, G)$ constituted

- either of $Q = \epsilon$ and $G = \top$ or $G = \bot$,
- either of a binder $Q = q_1x_1 \ldots q_nx_n$, $n > 0$, and a sequence of pairs of formulae $G = (P_1, N_1); \ldots; (P_n, N_n)$ such that the formulae $P_k$ and $N_k$ are only built on the propositional symbols $\{x_1, \ldots, x_{k-1}\}$ (or $\top$ or $\bot$ when $k = 1$).
Literal base

A literal base is a pair \((Q, G)\) constituted
- either of \(Q = \epsilon\) and \(G = \top\) or \(G = \bot\),
- either of a binder \(Q = q_1 x_1 \ldots q_n x_n, n > 0\), and a sequence of pairs of formulae \(G = (P_1, N_1); \ldots; (P_n, N_n)\) such that the formulae \(P_k\) and \(N_k\) are only built on the propositional symbols \(\{x_1, \ldots, x_{k-1}\}\) (or \(\top\) or \(\bot\) when \(k = 1\)).

Interpretation function

The interpretation function is defined as follows:
**Literal base**

A literal base is a pair \((Q, G)\) constituted

- either of \(Q = \epsilon\) and \(G = \top\) or \(G = \bot\),
- either of a binder \(Q = q_1x_1 \ldots q_nx_n, \ n > 0\), and a sequence of pairs of formulae \(G = (P_1, N_1); \ldots; (P_n, N_n)\) such that the formulae \(P_k\) and \(N_k\) are only built on the propositional symbols \(\{x_1, \ldots, x_{k-1}\}\) (or \(\top\) or \(\bot\) when \(k = 1\)).

**Interpretation function**

The interpretation function is defined as follows:

- if \(lb = (\epsilon, G)\) then \(lb^* = G\);
A unified framework for Certificate and Compilation for QBF

Literal base
A literal base is a pair \((Q, G)\) constituted
- either of \(Q = \epsilon\) and \(G = \top\) or \(G = \bot\),
- either of a binder \(Q = q_1x_1 \ldots q_nx_n, \ n > 0\), and a sequence of pairs of formulae \(G = (P_1, N_1); \ldots; (P_n, N_n)\) such that the formulae \(P_k\) and \(N_k\) are only built on the propositional symbols \(\{x_1, \ldots, x_{k-1}\}\) (or \(\top\) or \(\bot\) when \(k = 1\)).

Interpretation function
The interpretation function is defined as follows:
- if \(lb = (\epsilon, G)\) then \(lb^* = G\);
- if \(lb = (q_1x_1 \ldots q_nx_n, (P_1, N_1); \ldots; (P_n, N_n)), \ n > 0\), then

\[
lb^* = q_1x_1 \ldots q_nx_n \bigwedge_{1 \leq i \leq n} ((\neg x_i \lor P_i) \land (x_i \lor N_i))
\]
Optimality of a literal base

**Substitution following a literal base**

A substitution \([x_1 \leftarrow C_1] \ldots [x_p \leftarrow C_p]\) follows a literal base 
\((q_1 x_1 \ldots q_n x_n, (P_1, N_1); \ldots ; (P_n, N_n)), p < n\), if for all \(k\), 
\(1 \leq k \leq p + 1\)

- if \(C_k = \top\) then \(\models [x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](P_k)\)
- else \(\models [x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](N_k)\).
Optimality of a literal base

Substitution following a literal base

A substitution $[x_1 \leftarrow C_1] \ldots [x_p \leftarrow C_p]$ follows a literal base $(q_1x_1 \ldots q_nx_n, (P_1, N_1); \ldots; (P_n, N_n))$, $p < n$, if for all $k$, $1 \leq k \leq p + 1$

if $C_k = \top$ then $[x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](P_k)$
else $[x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](N_k)$.

$$lb = (\forall c \exists a \exists b,$$
$$ (P_c = \top, N_c = \top); (P_a = c, N_a = \top); (P_b = c, N_b = (a \leftrightarrow c)))$$
Optimality of a literal base

Substitution following a literal base

A substitution \([x_1 \leftarrow C_1] \ldots [x_p \leftarrow C_p]\) follows a literal base 
\((q_1x_1 \ldots q_n x_n, (P_1, N_1); \ldots; (P_n, N_n)), \ p < n\), if for all \(k\), 
\(1 \leq k \leq p + 1\)

if \(C_k = \top\) then \(\models [x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](P_k)\)
else \(\models [x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](N_k)\).

\[\]

- \(lb = (\forall c \exists a \exists b,\)
  
  \((P_c = \top, N_c = \top); (P_a = c, N_a = \top); (P_b = c, N_b = (a \leftarrow c)))\)

- Substitution \([c \leftarrow \top][a \leftarrow \bot]\) follows \(lb\) since \(\models P_c\) and
  \(\models [c \leftarrow \top](N_a)\).
Optimality of a literal base

Substitution following a literal base

A substitution \([x_1 \leftarrow C_1] \ldots [x_p \leftarrow C_p]\) follows a literal base \((q_1x_1 \ldots q_nx_n, (P_1, N_1); \ldots; (P_n, N_n))\), \(p < n\), if for all \(k\), \(1 \leq k \leq p + 1\)

- if \(C_k = \top\) then \(\models [x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](P_k)\)
- else \(\models [x_1 \leftarrow C_1] \ldots [x_{k-1} \leftarrow C_{k-1}](N_k)\).

\[
lb = (\forall c \exists a \exists b, (P_c = \top, N_c = \top); (P_a = c, N_a = \top); (P_b = c, N_b = (a \leftarrow c)))
\]

- Substitution \([c \leftarrow \top][a \leftarrow \bot]\) follows \(lb\) since \(\models P_c\) and \(\models [c \leftarrow \top](N_a)\),

- but substitution \([c \leftarrow \top][a \leftarrow \bot][b \leftarrow \bot]\) does not follow \(lb\) since \(\not\models [c \leftarrow \top][a \leftarrow \bot](N_b)\).
Optimality of a literal base

The literal base \( lb = (q_1 x_1 \ldots q_n x_n, (P_1, N_1); \ldots; (P_n, N_n)) \) such that \( lb^* = q_1 x_1 \ldots q_n x_n G \) is optimal if the following holds. For all \( i, 1 \leq i \leq n \), let \([x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}]\) be a substitution which follows \((P_1, N_1); \ldots; (P_{i-1}, N_{i-1})\)

Then

\[
\models [x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}](P_i)
\]

if and only if there exists a model for

\[
q_{i+1} x_{i+1} \ldots q_n x_n[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \top](G)
\]

and

\[
\models [x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}](N_i)
\]

if and only if there exists a model for

\[
q_{i+1} x_{i+1} \ldots q_n x_n[x_1 \leftarrow C_1] \ldots [x_{i-1} \leftarrow C_{i-1}][x_i \leftarrow \bot](G).
\]
\[ lb = (\forall c \exists a \exists b, (T, T); (c, T); (c, (a \leftrightarrow c))) \]
$lb = (\forall c \exists a \exists b, (\top, \top); (c, \top); (c, (a \leftrightarrow c)))$

- $lb$ is an optimal literal base such that
  $lb^* \equiv \forall c \exists a \exists b((b \lor a) \leftrightarrow c)$. 
$lb = (\forall c \exists a \exists b, (\top, \top); (c, \top); (c, (a \leftrightarrow c)))$

- $lb$ is an optimal literal base such that $lb^* \equiv \forall c \exists a \exists b((b \lor a) \leftrightarrow c)$.
- $\hat{a}_c = c$ and $\hat{b}_c = c$ is a model for $lb^*$. 
$lb = (\forall c \exists a \exists b, (T, T); (c, T); (c, (a \leftrightarrow c)))$

- $lb$ is an optimal literal base such that $lb^* \cong \forall c \exists a \exists b((b \lor a) \leftrightarrow c)$.

- $\hat{a}_c = c$ and $\hat{b}_c = c$ is a model for $lb^*$.
  - Substitution $[c \leftarrow T][a \leftarrow T]$ is in the QBF model and follows the literal base,
$lb = (\forall c \exists a \exists b, (\top, \top); (c, \top); (c, (a \leftrightarrow c)))$

- $lb$ is an optimal literal base such that $lb^* \equiv \forall c \exists a \exists b((b \lor a) \leftrightarrow c)$.
- $\hat{a}_c = c$ and $\hat{b}_c = c$ is a model for $lb^*$.
  - Substitution $[c \leftarrow \top][a \leftarrow \top]$ is in the QBF model and follows the literal base,
  - Next move from the model for the $\exists$-player is $\hat{b}_c(\top) = \top$, 
\[ lb = (\forall c \exists a \exists b, (\top, \top); (c, \top); (c, (a \leftrightarrow c))) \]

- \( lb \) is an optimal literal base such that
  \[ lb^* \cong \forall c \exists a \exists b((b \lor a) \leftrightarrow c). \]

- \( \hat{a}_c = c \) and \( \hat{b}_c = c \) is a model for \( lb^* \).
  - Substitution \([c \leftarrow \top][a \leftarrow \top]\) is in the QBF model and follows the literal base,
  - Next move from the model for the \( \exists \)-player is \( \hat{b}_c(\top) = \top \),
  - Since \( \models [c \leftarrow \top][a \leftarrow \top][(a \leftrightarrow c)] : \exists \)-player may change his mind.
$lb = (\forall c \exists a \exists b, (T, T); (c, T); (c, (a \leftrightarrow c)))$

- $lb$ is an optimal literal base such that $lb^* \cong \forall c \exists a \exists b((b \lor a) \leftrightarrow c)$.
- $\hat{a}_c = c$ and $\hat{b}_c = c$ is a model for $lb^*$.
  - Substitution $[c \leftarrow T][a \leftarrow T]$ is in the QBF model and follows the literal base,
  - Next move from the model for the $\exists$-player is $\hat{b}_c(T) = T$,
  - Since $\models [c \leftarrow T][a \leftarrow T]((a \leftrightarrow c)) : \exists$-player may change his mind.
- $\hat{a}_c = \bot$ and $\hat{b}_c = c$ is a model for $lb^*$.
\[ lb = (\forall c \exists a \exists b, (\top, \top); (c, \top); (c, (a \leftrightarrow c))) \]

- \( lb \) is an optimal literal base such that
  \( lb^* \cong \forall c \exists a \exists b((b \lor a) \leftrightarrow c) \).

- \( \hat{a}_c = c \) and \( \hat{b}_c = c \) is a model for \( lb^* \).
  - Substitution \([c \leftarrow \top][a \leftarrow \top]\) is in the QBF model and follows the literal base,
  - Next move from the model for the \( \exists \)-player is \( \hat{b}_c(\top) = \top \),
  - Since \( \models [c \leftarrow \top][a \leftarrow \top]((a \leftrightarrow c)) : \exists \)-player may change his mind.

- \( \hat{a}_c = \bot \) and \( \hat{b}_c = c \) is a model for \( lb^* \).
  - Substitution \([c \leftarrow \bot][a \leftarrow \bot]\) is in the QBF model and follows the literal base,
$lb = (\forall c \exists a \exists b, (T, T); (c, T); (c, (a \leftrightarrow c)))$

- $lb$ is an optimal literal base such that $lb^* \cong \forall c \exists a \exists b((b \lor a) \leftrightarrow c)$.
- $\hat{a}_c = c$ and $\hat{b}_c = c$ is a model for $lb^*$.
  - Substitution $[c \leftarrow T][a \leftarrow T]$ is in the QBF model and follows the literal base,
  - Next move from the model for the $\exists$-player is $\hat{b}_c(T) = T$,
  - Since $\models [c \leftarrow T][a \leftarrow T]((a \leftrightarrow c))$ : $\exists$-player may change his mind.
- $\hat{a}_c = \bot$ and $\hat{b}_c = c$ is a model for $lb^*$.
  - Substitution $[c \leftarrow \bot][a \leftarrow \bot]$ is in the QBF model and follows the literal base,
  - Next move from the model for the $\exists$-player is $\hat{b}_c(T) = \bot$, 

$lb = (\forall c \exists a \exists b, (\top, \top); (c, \top); (c, (a \leftarrow c)))$

- $lb$ is an optimal literal base such that $lb^* \equiv \forall c \exists a \exists b((b \lor a) \leftarrow c)$.
- $\hat{a}_c = c$ and $\hat{b}_c = c$ is a model for $lb^*$.
  - Substitution $[c \leftarrow \top][a \leftarrow \top]$ is in the QBF model and follows the literal base,
  - Next move from the model for the $\exists$-player is $\hat{b}_c(\top) = \top$,
  - Since $\models [c \leftarrow \top][a \leftarrow \top]((a \leftarrow c)) : \exists$-player may change his mind.
- $\hat{a}_c = \bot$ and $\hat{b}_c = c$ is a model for $lb^*$.
  - Substitution $[c \leftarrow \bot][a \leftarrow \bot]$ is in the QBF model and follows the literal base,
  - Next move from the model for the $\exists$-player is $\hat{b}_c(\top) = \bot$,
  - Since $\not\models [c \leftarrow \bot][a \leftarrow \bot](c) : \exists$-player cannot change his mind.
Theorem

The next move choice problem is polytime for the interpretation of an optimal literal base.
Theorem

The next move choice problem is polytime for the interpretation of an optimal literal base.

Minimality of a QBF

A QBF is minimal if all the (propositional) models of the matrix are (at least) in one of its (QBF) model.
Theorem

The next move choice problem is polytime for the interpretation of an optimal literal base.

Minimality of a QBF

A QBF is minimal if all the (propositional) models of the matrix are (at least) in one of its (QBF) model.

Theorem

The interpretation of an optimal literal base is minimal.
An operator to combine compiled QBF

⊕ operator

\[ Q' = q_2 x_2 \ldots q_n x_n \text{ and } Q = q_1 x_1 Q' \]
\[ (Q, (P_1, N_1); \ldots; (P_n, N_n)) \oplus (Q, (P'_1, N'_1); \ldots; (P'_n, N'_n)) = (Q, ((P_1 \lor P'_1), (N_1 \lor N'_1)); \]
\[ (P_2 \land (P'_2 \lor x) \land (P_2 \lor x'), N_2 \land (N'_2 \lor x) \land (N_2 \lor x')); \]
\[ (P_n \land (P'_n \lor x) \land (P_n \lor x'), N_n \land (N'_n \lor x) \land (N_n \lor x'))) \]

with \( x = ((\neg x_1 \lor P_1) \land (x_1 \lor N_1)), x' = ((\neg x_1 \lor P'_1) \land (x_1 \lor N'_1)) \) and

\[ (Q', (P_2, N_2); \ldots; (P_n, N_n)) \oplus (Q', (P'_2, N'_2); \ldots; (P'_n, N'_n)) = (Q', (P_2, N_2); \ldots; (P_n, N_n)) \oplus (Q', (P'_2, N'_2); \ldots; (P'_n, N'_n)) \]
Theorem

If $lb^* = QM$, $lb'^* = QM'$ and $(lb \oplus lb')^* = QM_\oplus$
then $M_\oplus \equiv (M \lor M')$. 
Theorem  

If \( lb^* = QM, \ lb'^* = QM' \) and \( (lb \oplus lb')^* = QM_\oplus \)  
then \( M_\oplus \equiv (M \lor M') \).

Theorem  

If \( lb = (Q, G) \) such that \( lb^* \cong Q[x \leftarrow T](M) \) and \( lb' = (Q, G') \) such that \( lb'^* \cong Q[x \leftarrow \bot](M) \)  
are optimal literal bases  
then \( lb_\oplus = ((q \times Q, (T, \bot); G) \oplus (q \times Q, (\bot, T); G')) \) is an optimal literal base such that \( lb^*_\oplus \cong q \times QM. \)
Compilation of QBF with a search-based algorithm

from search_comp_qbf :

\[ Q = q x Q' \]
\[ \text{lb}^+ := \text{search\_comp\_qbf}(Q', [x \leftarrow \top](M)) \]
\[ \text{lb}^- := \text{search\_comp\_qbf}(Q', [x \leftarrow \bot](M)) \]

if ((\(q = \forall\)) and ((\text{lb}^+ = \text{nonvalid}) or (\text{lb}^- = \text{nonvalid}))) or
((\(q = \exists\)) and ((\text{lb}^+ = \text{nonvalid}) and (\text{lb}^- = \text{nonvalid}))) then

return \(\text{nonvalid}\)

else if ((\(q = \exists\)) and (\text{lb}^+ = \text{nonvalid}) and (\text{lb}^- = (Q', G^-))) then

return (\(Q, (\bot, \top) ; G^-\))

else if ((\(q = \exists\)) and (\text{lb}^- = \text{nonvalid}) and (\text{lb}^+ = (Q', G^+))) then

return (\(Q, (\top, \bot) ; G^+\))

else

\(\text{lb}^+ = (Q', G^+)\) and \(\text{lb}^- = (Q', G^-)\)

return (\(Q, (\top, \bot) ; G^+\) \(\oplus\) (\(Q, (\bot, \top) ; G^-\))

and if

1. Stéphan ICLA 2009, Chennai, India
29/35
Compilation of QBF with a search-based algorithm

from search_comp_qbf:

\( Q = q \times Q' \)
\( l^+ := \text{search\_comp\_qbf}(Q', [x \leftarrow \top](M)) \)
\( l^- := \text{search\_comp\_qbf}(Q', [x \leftarrow \bot](M)) \)

if ((\( q = \forall \)) and ((\( l^+ = \text{nonvalid} \)) or ((\( l^- = \text{nonvalid} \)))) or
((\( q = \exists \)) and ((\( l^+ = \text{nonvalid} \)) and ((\( l^- = \text{nonvalid} \)))) ) then
  return \( \text{nonvalid} \)
else if ((\( q = \exists \)) and ((\( l^+ = \text{nonvalid} \)) and ((\( l^- = (Q', G^-) \))))
then
  return \(( Q, (\bot, \top) ; G^-) \)
else if ((\( q = \exists \)) and ((\( l^- = \text{nonvalid} \)) and ((\( l^+ = (Q', G^+) \))))
then
  return \(( Q, (\top, \bot) ; G^+) \)
else
  \( l^+ = (Q', G^+) \) and \( l^- = (Q', G^-) \)
  return \(( Q, (\top, \bot) ; G^+) \oplus (Q, (\bot, \top) ; G^-) \)
and if
from search_comp_qbf :

\[ Q = q \times Q' \]

\[ l_{b}^+ := \text{search\_comp\_qbf}(Q', [x \leftarrow \top](M)) \]

\[ l_{b}^- := \text{search\_comp\_qbf}(Q', [x \leftarrow \bot](M)) \]

\textbf{if } ((q = \forall) \text{ and } ((l_{b}^+ = \text{nonvalid}) \text{ or } (l_{b}^- = \text{nonvalid}))) \text{ or } ((q = \exists) \text{ and } ((l_{b}^+ = \text{nonvalid}) \text{ and } (l_{b}^- = \text{nonvalid}))) \text{ then}

\textbf{return } \text{nonvalid}

\textbf{else if } ((q = \exists) \text{ and } (l_{b}^+ = \text{nonvalid}) \text{ and } (l_{b}^- = (Q', G^-))) \text{ then}

\textbf{return } (Q, (\bot, \top); G^-)

\textbf{else if } ((q = \exists) \text{ and } (l_{b}^- = \text{nonvalid}) \text{ and } (l_{b}^+ = (Q', G^+))) \text{ then}

\textbf{return } (Q, (\top, \bot); G^+)

\textbf{else}

\[ l_{b}^+ = (Q', G^+) \text{ and } l_{b}^- = (Q', G^-) \]

\textbf{return } (Q, (\top, \bot); G^+) \oplus (Q, (\bot, \top); G^-)

\textbf{and if}
from search_comp_qbf:

\[ Q = q \times Q' \]

\[ lb^+ := search\_comp\_qbf(Q', [x \leftarrow \top](M)) \]

\[ lb^- := search\_comp\_qbf(Q', [x \leftarrow \bot](M)) \]

if ((q = \forall) and ((lb^+ = nonvalid) or (lb^- = nonvalid))) or ((q = \exists) and ((lb^+ = nonvalid) and (lb^- = nonvalid)) ) then

    return nonvalid

else if ((q = \exists) and (lb^+ = nonvalid) and (lb^- = (Q', G^-))) then

    return (Q, (\bot, \top) ; G^-)

else if ((q = \exists) and (lb^- = nonvalid) and (lb^+ = (Q', G^+))) then

    return (Q, (\top, \bot) ; G^+)

else

    lb^+ = (Q', G^+) and lb^- = (Q', G^-)

    return (Q, (\top, \bot) ; G^+) \oplus (Q, (\bot, \top) ; G^-)

and if
from search_comp_qbf:

\[ Q = q \times Q' \]

\[ lb^+ := \text{search\_comp\_qbf}(Q', [x \leftarrow \top](M)) \]

\[ lb^- := \text{search\_comp\_qbf}(Q', [x \leftarrow \bot](M)) \]

if ((q = \forall) and ((lb^+ = nonvalid) or (lb^- = nonvalid))) or
((q = \exists) and ((lb^+ = nonvalid) and (lb^- = nonvalid))) then

return nonvalid

else if ((q = \exists) and (lb^+ = nonvalid) and (lb^- = (Q', G^-))) then

return (Q, (\bot, \top) ; G^-)

else if ((q = \exists) and (lb^- = nonvalid) and (lb^+ = (Q', G^+))) then

return (Q, (\top, \bot) ; G^+)

else

lb^+ = (Q', G^+) and lb^- = (Q', G^-)

return (Q, (\top, \bot) ; G^+) \oplus (Q, (\bot, \top) ; G^-)

and if
Theorem

Let \( QM \) be a QBF.

\(\text{search\_comp\_qbf}(Q, M)\) returns a literal base \( lb \) such that

\( lb^* \cong QM \) if \( QM \) is valid and returns \text{non\_valid} otherwise.
Theorem

Let $QM$ be a QBF.

$\text{search\_comp\_qbf}(Q, M)$ returns a literal base $lb$ such that $lb^* \cong QM$ if $QM$ is valid and returns $\text{non\_valid}$ otherwise.

Theorem

Let $QM$ be a valid QBF.

Then $\text{search\_comp\_qbf}(Q, M)$ is an optimal literal base.
Conclusion
A unified framework for...
A unified framework for...

- the representation of certificates,
Conclusion

A unified framework for...

- the representation of certificates,
- a target language for compilation
Conclusion

A unified framework for...

- the representation of certificates,
- a target language for compilation

Two implementations

- one in Prolog with formulae
## Conclusion

**A unified framework for...**
- the representation of certificates,
- a target language for compilation

**Two implementations**
- one in Prolog with formulae
- one in C++ with functions represented by BDD using the CUDD library