Nested Boolean Functions as Models for QBFs

Uwe Bubeck and Hans Kleine Büning

University of Paderborn

July 11, 2013

Outline

- Introduction: QBF and (Counter-)Models
- Free Variables and Models
- NBF Representation
- Conclusion

Introduction:
QBF and (Counter-)Models

Quantified Boolean Formulas

QBF extends propositional logic by allowing universal and existential quantifiers over propositional variables.

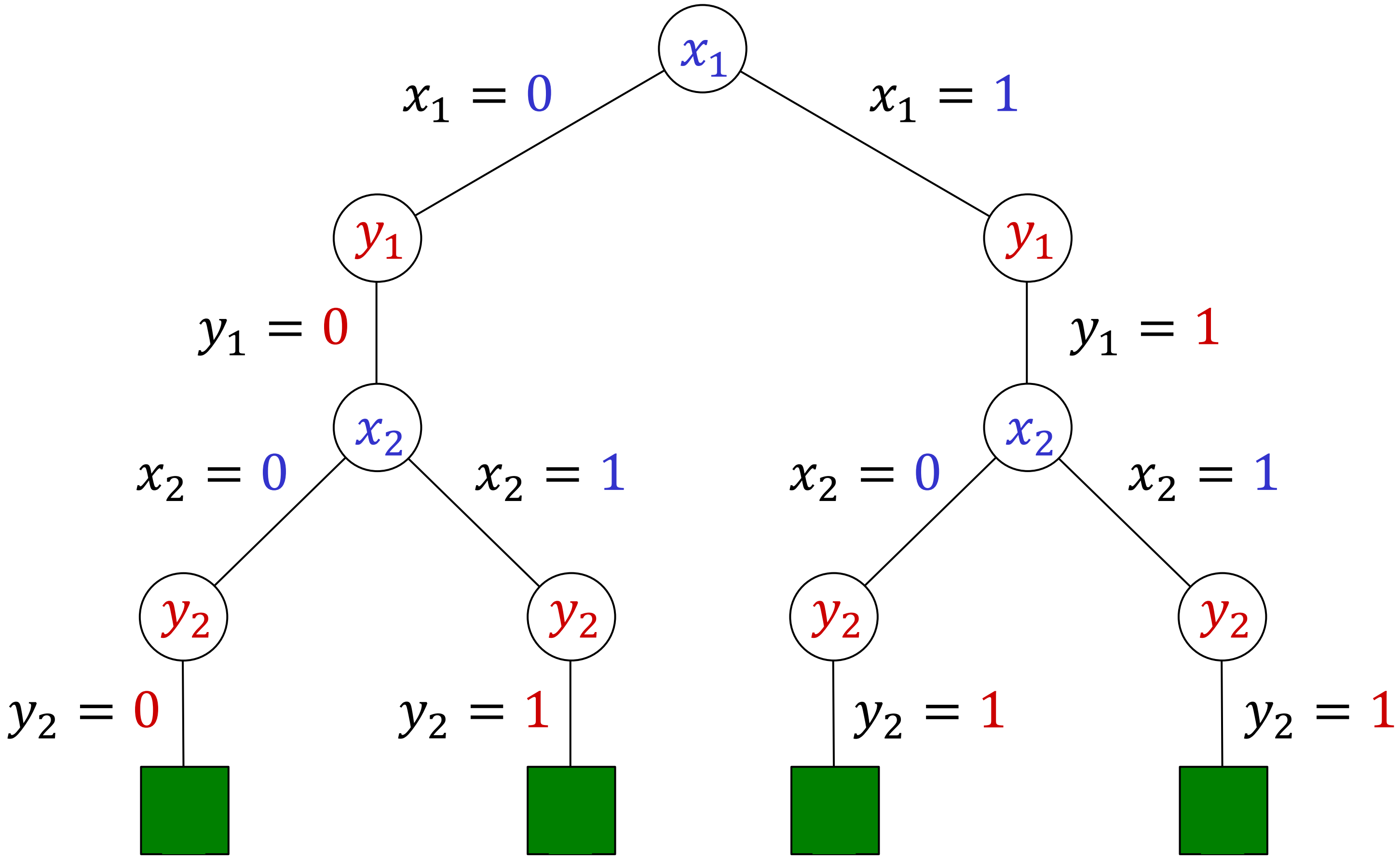
Semantics of closed QBF:

$\exists y\ \Phi(y)$ is true if and only if
$\Phi[y/0]$ is true or $\Phi[y/1]$ is true.

$\forall x\ \Phi(x)$ is true if and only if
$\Phi[x/0]$ is true and $\Phi[x/1]$ is true.

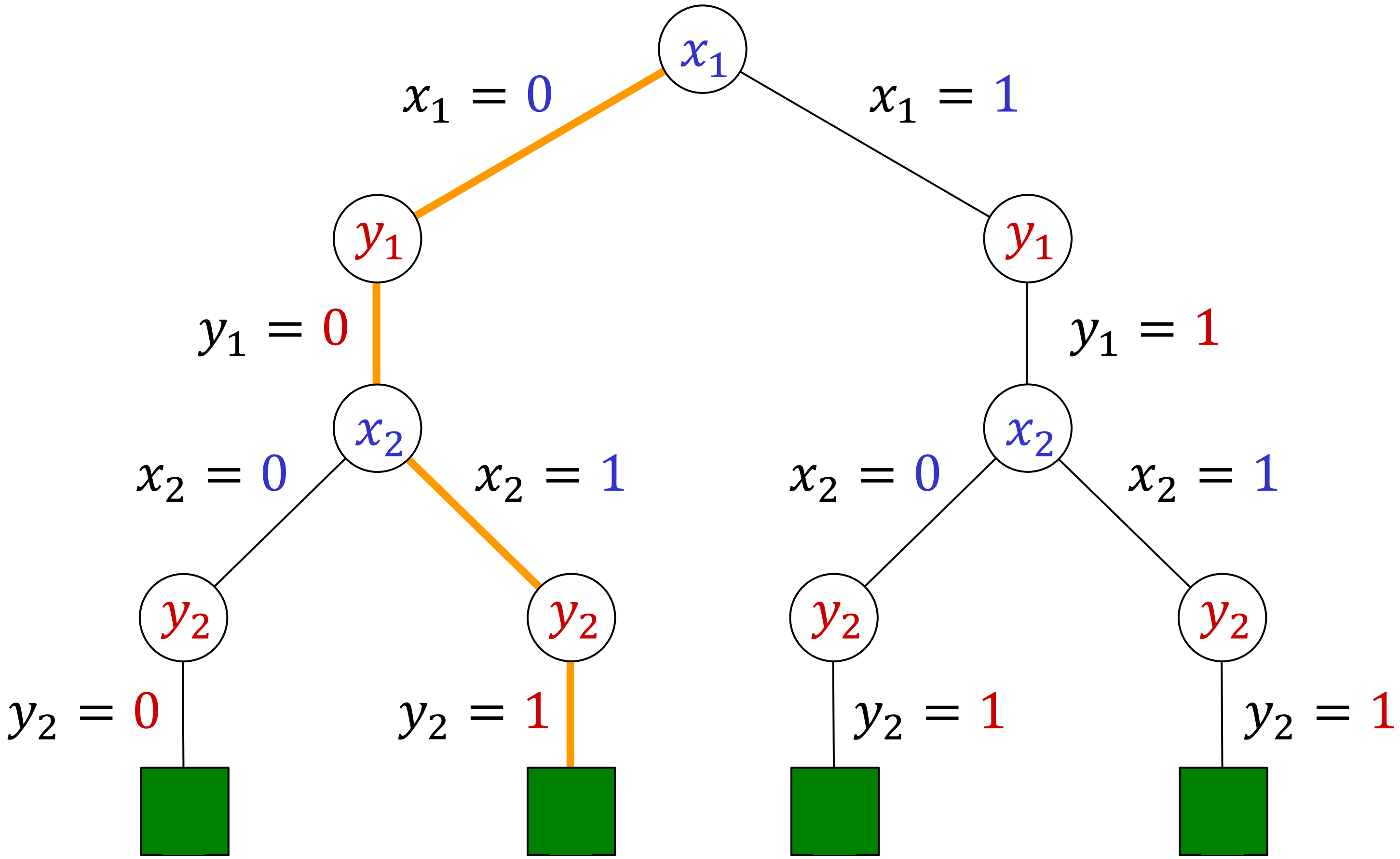
Tree Models

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2\ (x_1 \vee \neg y_1) \wedge (\neg x_1 \vee y_2) \wedge (y_1 \vee x_2 \vee \neg y_2) \wedge (\neg x_2 \vee y_2)$$



Function Models 1/2

QBF as a 2-player game: ∃ and ∀ player alternatingly choose assignments for variables in prefix order.



Function Models 2/2

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 \dots \forall x_n \exists y_n\ \phi(x_1, \dots, x_n, y_1, \dots, y_n) = \text{true}$$

if and only if

$$\forall x_1 \dots \forall x_n\ \phi(x_1, \dots, x_n, f_1(x_1), \dots, f_n(x_1, \dots, x_n)) = \text{true}$$

for some $f_1, \dots, f_n$ (Skolem functions).

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 \dots \forall x_n \exists y_n\ \phi(x_1, \dots, x_n, y_1, \dots, y_n) = \text{false}$$

if and only if

$$\exists y_1 \dots \exists y_n\ \phi(g_1(), g_2(y_1), \dots, g_n(y_1, \dots, y_{n-1}), y_1, \dots, y_n) = \text{false}$$

for some $g_1, \dots, g_n$ (Herbrand functions).

Motivation

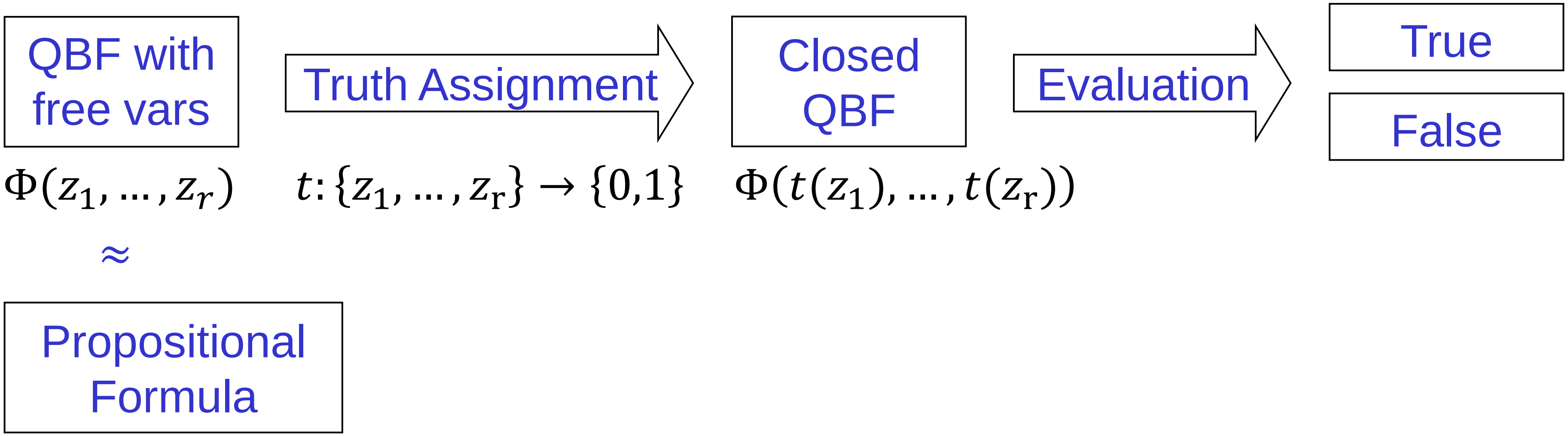
- Important applications: solver certificates, explanations, ...
- Balabanov and Jiang (2012):
  Extract Skolem model from cube-resolution proof,
  Herbrand countermodel from clause-resolution proof.
- Problem: compact representation
  (no polynomial-size propositional encoding if $\Sigma_2^P \neq \Pi_2^P$)
- Contributions:
  - direct polynomial-size encoding by NBFs
  - (counter)models parameterized by free variables

Free Variables and Models

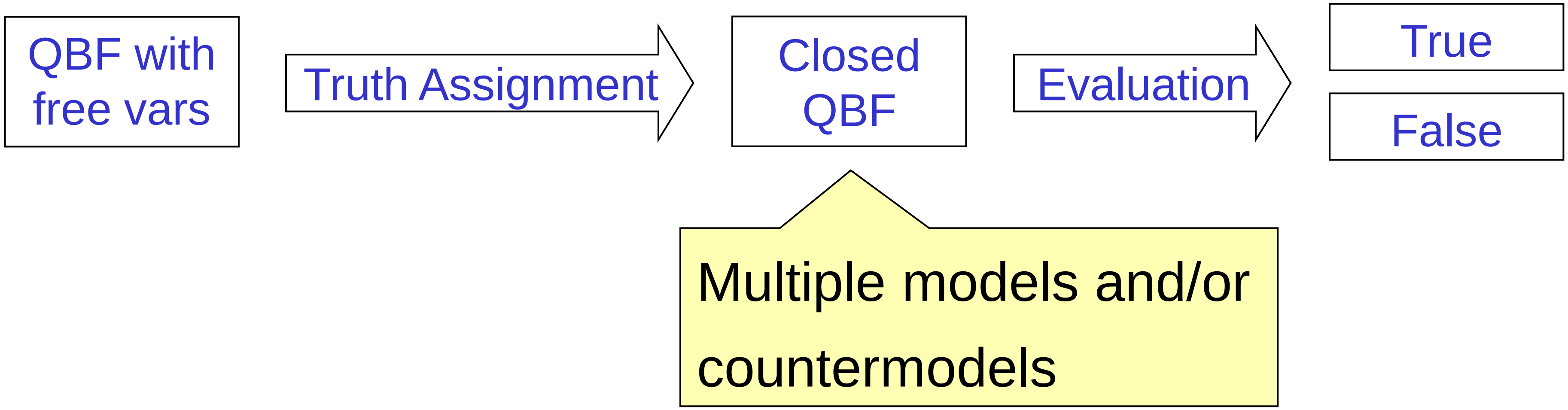
Semantics of Free Variables

Closed QBF: either true or false

Open QBF: valuation depends on the free variables:



Free Variables and Models



How are the models and countermodels for different assignments to the free variables related to each other?

Complete Equivalence Models 1/2

Idea: Replace all quantified variables with functions over the free variables.

$$\forall v_n \exists v_{n-1} \dots \forall v_2 \exists v_1 \; \phi(v_1, \dots, v_n, z_1, \dots, z_r)$$

$$\approx$$

$$\phi(h_1(z_1, \dots, z_r), \dots, h_n(z_1, \dots, z_r), z_1, \dots, z_r)$$

Complete Equivalence Models 2/2

Why bother about models parameterized by free variables?

Non-prenex QBF:

$$\forall a \exists b \boxed{\left(\forall c \exists d\ \alpha(a,b,c,d)\right)} \wedge \left(\forall x \exists y\ \beta(a,b,x,y)\right)$$

Open QBF with

free vars a, b.

e.g. precompute partial certificate.

NBF Representation

Nested Boolean Functions

A Nested Boolean Function (NBF) [Cook/Soltys 1999] is a sequence of functions $F = (f_0, \dots, f_k)$ with

- initial functions $f_0, \dots, f_t$ given as propositional formulas
- compound functions $f_i(x^i) := f_{j_0}(f_{j_1}(x_1^i), \dots, f_{j_r}(x_r^i))$ for previously defined functions $f_{j_0}, \dots, f_{j_r}$.

Example: parity of Boolean variables

$$f_0(p_1, p_2) := (\neg p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2)$$

$$f_1(p_1, p_2, p_3, p_4) := f_0(f_0(p_1, p_2), f_0(p_3, p_4))$$

$$f_2(p_1, \dots, p_{16}) := f_1(f_1(p_1, \dots, p_4), \dots, f_1(p_{13}, \dots, p_{16}))$$

Quantifier Encoding in NBF 1/2

QBF: $\Phi(\boldsymbol{z}) \coloneqq \exists x \; \phi(x, \boldsymbol{z})$

$$\Phi(\boldsymbol{z}) \approx F_1(\boldsymbol{z})$$

NBF:

$$F_0(x, \boldsymbol{z}) \coloneqq \phi(x, \boldsymbol{z})$$

$$F_1(\boldsymbol{z}) \coloneqq F_0(\underbrace{F_0(1, \boldsymbol{z})}_{=1 \text{ if } x=1 \text{ is a satisfying choice}}, \boldsymbol{z})$$

$$\underbrace{F_0(F_0(1, \boldsymbol{z}), \boldsymbol{z})} = F_0(1, \boldsymbol{z}) = 1$$

Quantifier Encoding in NBF 1/2

QBF: $\Phi(\boldsymbol{z}) := \exists x\ \phi(x, \boldsymbol{z})$

NBF:

$$F_0(x, \boldsymbol{z}) := \phi(x, \boldsymbol{z})$$

$$F_1(\boldsymbol{z}) := \underbrace{F_0(\underbrace{F_0(1, \boldsymbol{z})}_{= 0 \text{ if } x = 1 \text{ is not satisfying}}, \boldsymbol{z})}_{= F_0(0, \boldsymbol{z})}$$

Quantifier Encoding in NBF 2/2

QBF: $\Phi(\boldsymbol{z}) := \forall y \exists x \; \phi(x, y, \boldsymbol{z})$

NBF:

$$F_0(x, y, \boldsymbol{z}) := \phi(x, y, \boldsymbol{z})$$

$$F_1(y, \boldsymbol{z}) := F_0(F_0(1, y, \boldsymbol{z}), y, \boldsymbol{z})$$

$$F_2(\boldsymbol{z}) := \underbrace{F_1(\underbrace{F_1(0, \boldsymbol{z})}_{= 0 \text{ if } y = 0 \text{ is not satisfying}}, \boldsymbol{z})}_{= F_1(0, \boldsymbol{z}) = 0}$$

Quantifier Encoding in NBF 2/2

QBF: $\Phi(\boldsymbol{z}) \coloneqq \forall y \exists x \, \phi(x, y, \boldsymbol{z})$

NBF:

$$F_0(x, y, \boldsymbol{z}) \coloneqq \phi(x, y, \boldsymbol{z})$$
$$F_1(y, \boldsymbol{z}) \coloneqq F_0(F_0(1, y, \boldsymbol{z}), y, \boldsymbol{z})$$
$$F_2(\boldsymbol{z}) \coloneqq \underbrace{F_1(\underbrace{F_1(0, \boldsymbol{z})}_{= 1 \text{ if } y = 0 \text{ is satisfying}}, \boldsymbol{z})}_{= F_1(1, \boldsymbol{z})}$$

Quantifier Encoding in NBF 2/2

QBF: $\Phi(\boldsymbol{z}) \coloneqq \forall y \exists x \; \phi(x, y, \boldsymbol{z})$

NBF:

$$\begin{aligned} F_0(x, y, \boldsymbol{z}) &\coloneqq \phi(x, y, \boldsymbol{z}) \\ F_1(y, \boldsymbol{z}) &\coloneqq F_0(F_0(1, y, \boldsymbol{z}), y, \boldsymbol{z}) \\ F_2(\boldsymbol{z}) &\coloneqq F_1(F_1(0, \boldsymbol{z}), \boldsymbol{z}) \end{aligned}$$

→ Concise representation of QDPLL branching:
innermost call of F_i is the first branch,
outermost call of F_i is the second branch, or a repetition of the first one if it is already conclusive.

Complete Equiv. Model in NBF 1/2

First branch determines which branch is conclusive.

→ this is our witness, i.e. (counter)model.

Complete Equiv. Model in NBF 1/2

First branch determines which branch is conclusive.

→ this is our witness, i.e. (counter)model.

QBF: $\Phi(\boldsymbol{z}) \coloneqq \forall y \exists x \; \phi(x, y, \boldsymbol{z})$

NBF:

$$F_0(x, y, \boldsymbol{z}) \coloneqq \phi(x, y, \boldsymbol{z})$$

$$F_1(y, \boldsymbol{z}) \coloneqq F_0(\underbrace{F_0(1, y, \boldsymbol{z})}_{h_x(\boldsymbol{z})}, y, \boldsymbol{z})$$

$$F_2(\boldsymbol{z}) \coloneqq F_1(\underbrace{F_1(0, \boldsymbol{z})}_{h_y(\boldsymbol{z})}, \boldsymbol{z})$$

Complete Equiv. Model in NBF 1/2

QBF: $\Phi(\boldsymbol{z}) \coloneqq \forall y \exists x\ \phi(x, y, \boldsymbol{z})$

$$F_0(x, y, \boldsymbol{z}) \coloneqq \phi(x, y, \boldsymbol{z})$$

NBF:

$$F_1(y, \boldsymbol{z}) \coloneqq F_0\Big(F_0\big(1, h_y(\boldsymbol{z}), \boldsymbol{z}\big), y, \boldsymbol{z}\Big)$$

$$F_2(\boldsymbol{z}) \coloneqq F_1\big(F_1(0, \boldsymbol{z}), \boldsymbol{z}\big)$$

$F_0\big(1, h_y(\boldsymbol{z}), \boldsymbol{z}\big) \rightarrow h_x(\boldsymbol{z})$

$F_1(0, \boldsymbol{z}) \rightarrow h_y(\boldsymbol{z})$

Complete Equiv. Model in NBF 2/2

In general:

$$\Phi(\boldsymbol{z}) \coloneqq Q_n v_n \dots Q_1 v_1\ \phi(v_1, \dots, v_n, \boldsymbol{z})$$

Complete equivalence model:

$$h_i(\boldsymbol{z}) \ \coloneqq \begin{cases} F_{i-1}(0, h_{i+1}(\boldsymbol{z}), \dots, h_1(\boldsymbol{z}), \boldsymbol{z}) & \text{, if } Q_i = \forall \\ F_{i-1}(1, h_{i+1}(\boldsymbol{z}), \dots, h_1(\boldsymbol{z}), \boldsymbol{z}) & \text{, if } Q_i = \exists \end{cases}$$

Clearly polynomial size, which is not possible with a propositional encoding if $\Sigma_2^P \neq \Pi_2^P$.

Admittedly more difficult to evaluate. But:

Equiv. model checking PSPACE-hard even if $h_i(\boldsymbol{z}) \in \{0,1\}$.

Conclusion

Conclusion

- Complete equivalence models as a generalization of Skolem/Herbrand (counter)models parameterized by free variables.
- Concise characterization of QDPLL branching and thus polynomial space by nested Boolean functions with one initial function and special recursive instantiation.

Future Work

- Restrictions on the model structure for subclasses of QBF, e.g. Horn, 2-CNF, etc.
- Build a NBF solver.

The End