universität innsbruck

Constraint Solving

René Thiemann and Fabian Mitterwallner

based on a previous course by Aart Middeldorp

Outline

1. **Nelson–Oppen Combination Method**
2. **Quantified Boolean Formulas**
3. **PSPACE-Completeness of QBF**
4. **Further Reading**

Current State

- SMT solver for various theories

Current State

- SMT solver for various theories
  - equality logic — equality graphs
  - EUF (equality with uninterpreted functions) — congruence closure
  - LRA (linear rational arithmetic) — simplex
  - LIA (linear integer arithmetic) — branch and bound
  - BV (bit-vector arithmetic) — flattening

Current State

- SMT solver for various theories
  - equality logic — equality graphs
  - EUF (equality with uninterpreted functions) — congruence closure
  - LRA (linear rational arithmetic) — simplex
  - LIA (linear integer arithmetic) — branch and bound
  - BV (bit-vector arithmetic) — flattening
- missing: combination of theories, e.g., EUF + LRA?

$$f(x+1) < f(y) \wedge f(f(x)) \neq f(x) - 1$$

Current State

- SMT solver for various theories
  - equality logic — equality graphs
  - EUF (equality with uninterpreted functions) — congruence closure
  - LRA (linear rational arithmetic) — simplex
  - LIA (linear integer arithmetic) — branch and bound
  - BV (bit-vector arithmetic) — flattening
- missing: combination of theories, e.g., EUF + LRA?

$$f(x+1) < f(y) \wedge f(f(x)) \neq f(x) - 1$$

- observation: all theories support equality

Current State

- SMT solver for various theories
  - equality logic — equality graphs
  - EUF (equality with uninterpreted functions) — congruence closure
  - LRA (linear rational arithmetic) — simplex
  - LIA (linear integer arithmetic) — branch and bound
  - BV (bit-vector arithmetic) — flattening
- missing: combination of theories, e.g., EUF + LRA?

$$f(x+1) < f(y) \wedge f(f(x)) \neq f(x) - 1$$

- observation: all theories support equality
- upcoming: combination method which communicate via equalities

Current State

- SMT solver for various theories
  - equality logic — equality graphs
  - EUF (equality with uninterpreted functions) — congruence closure
  - LRA (linear rational arithmetic) — simplex
  - LIA (linear integer arithmetic) — branch and bound
  - BV (bit-vector arithmetic) — flattening
- missing: combination of theories, e.g., EUF + LRA?

$$f(x+1) < f(y) \wedge f(f(x)) \neq f(x) - 1$$

- observation: all theories support equality
- upcoming: combination method which communicate via equalities
- requirement for presented combination algorithm: theories must be stably infinite

Definition

theory is stably infinite if every satisfiable quantifier-free formula has infinite model

Definition

theory is stably infinite if every satisfiable quantifier-free formula has infinite model

Examples

- EUF is stably infinite

Definition

theory is stably infinite if every satisfiable quantifier-free formula has infinite model

Examples

- EUF is stably infinite
- LIA and LRA are stably infinite

Definition

theory is stably infinite if every satisfiable quantifier-free formula has infinite model

Examples

- EUF is stably infinite
- LIA and LRA are stably infinite
- theory $T = (\Sigma, \mathcal{A})$ with $\Sigma = \{\mathsf{a}, \mathsf{b}, =\}$ and $\mathcal{A} = \{\forall x.\ x = \mathsf{a} \vee x = \mathsf{b}\}$ is not stably infinite

Definition

theory is stably infinite if every satisfiable quantifier-free formula has infinite model

Examples

- EUF is stably infinite
- LIA and LRA are stably infinite
- theory $T = (\Sigma, \mathcal{A})$ with $\Sigma = \{a, b, =\}$ and $\mathcal{A} = \{\forall x.\ x = a \lor x = b\}$ is not stably infinite
- BV is not stably infinite

Definition

(first-order) theory consists of

- signature Σ: set of function and predicate symbols
- axioms T: set of sentences in first-order logic in which only function and predicate symbols of Σ appear

Definition

(first-order) theory consists of

- signature Σ: set of function and predicate symbols
- axioms T: set of sentences in first-order logic in which only function and predicate symbols of Σ appear

Definition

theory combination $T_1 \oplus T_2$ of two theories

- T_1 over signature Σ_1
- T_2 over signature Σ_2

has signature $\Sigma_1 \cup \Sigma_2$ and axioms $T_1 \cup T_2$

Outline

Example

combination of linear arithmetic and uninterpreted functions:

$$x \geqslant y \ \wedge \ y - z \geqslant x \ \wedge \ \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \ \wedge \ z \geqslant 0$$

Example

combination of linear arithmetic and uninterpreted functions:

$$x \geqslant y \ \wedge\ y - z \geqslant x \ \wedge\ \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \ \wedge\ z \geqslant 0$$

Assumptions

two stably infinite theories

- T_1 over signature Σ_1
- T_2 over signature Σ_2

Example

combination of linear arithmetic and uninterpreted functions:

$$x \geqslant y \ \wedge \ y - z \geqslant x \ \wedge \ \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \ \wedge \ z \geqslant 0$$

Assumptions

two stably infinite theories

- T_1 over signature Σ_1
- T_2 over signature Σ_2

such that

- $\Sigma_1 \cap \Sigma_2 = \{=\}$

Example

combination of linear arithmetic and uninterpreted functions:

$$x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge\; \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \;\wedge\; z \geqslant 0$$

Assumptions

two stably infinite theories

- T_1 over signature Σ_1
- T_2 over signature Σ_2

such that

- $\Sigma_1 \cap \Sigma_2 = \{=\}$
- T_1-satisfiability of quantifier-free Σ_1-formulas is decidable
- T_2-satisfiability of quantifier-free Σ_2-formulas is decidable

Nelson–Oppen Method: Nondeterministic Version

input: quantifier-free conjunction φ in theory combination $T_1 \oplus T_2$

output: satisfiable or unsatisfiable

Nelson–Oppen Method: Nondeterministic Version

input: quantifier-free conjunction φ in theory combination $T_1 \oplus T_2$

output: satisfiable or unsatisfiable

1. purification

$$\varphi \approx \varphi_1 \wedge \varphi_2 \quad \text{for } \Sigma_1\text{-formula } \varphi_1 \text{ and } \Sigma_2\text{-formula } \varphi_2$$

Example

formula φ in combination of LIA and EUF:

$$1 \leqslant x \ \wedge\ x \leqslant 2 \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(1) \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(2)$$

Example

formula φ in combination of LIA and EUF:

$$1 \leqslant x \ \wedge \ x \leqslant 2 \ \wedge \ \mathsf{f}(x) \neq \mathsf{f}(1) \ \wedge \ \mathsf{f}(x) \neq \mathsf{f}(2)$$

Example

formula φ in combination of LIA and EUF:

$$1 \leqslant x \;\wedge\; x \leqslant 2 \;\wedge\; \mathsf{f}(x) \neq \mathsf{f}(y) \;\wedge\; \mathsf{f}(x) \neq \mathsf{f}(2) \;\wedge\; y = 1$$

Example

formula φ in combination of LIA and EUF:

$$1 \leqslant x \ \wedge\ x \leqslant 2 \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(y) \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(2) \ \wedge\ y = 1$$

Example

formula φ in combination of LIA and EUF:

$$1 \leqslant x \ \wedge\ x \leqslant 2 \ \wedge\ \mathrm{f}(x) \neq \mathrm{f}(y) \ \wedge\ \mathrm{f}(x) \neq \mathrm{f}(z) \ \wedge\ y = 1 \ \wedge\ z = 2$$

Example

formula φ in combination of LIA and EUF:

$$\underbrace{1 \leqslant x \ \wedge \ x \leqslant 2 \ \wedge \ y = 1 \ \wedge \ z = 2}_{\varphi_1} \wedge \underbrace{\mathsf{f}(x) \neq \mathsf{f}(y) \wedge \mathsf{f}(x) \neq \mathsf{f}(z)}_{\varphi_2}$$

Nelson–Oppen Method: Nondeterministic Version

input: quantifier-free conjunction φ in theory combination $T_1 \oplus T_2$

output: satisfiable or unsatisfiable

1. purification

$$\varphi \ \approx \ \varphi_1 \wedge \varphi_2 \quad \text{for } \Sigma_1\text{-formula } \varphi_1 \text{ and } \Sigma_2\text{-formula } \varphi_2$$

2. guess
   - V is set of shared variables in φ_1 and φ_2

Nelson–Oppen Method: Nondeterministic Version

input: quantifier-free conjunction φ in theory combination $T_1 \oplus T_2$

output: satisfiable or unsatisfiable

1. purification

$$\varphi \approx \varphi_1 \wedge \varphi_2 \quad \text{for } \Sigma_1\text{-formula } \varphi_1 \text{ and } \Sigma_2\text{-formula } \varphi_2$$

2. guess
   - V is set of shared variables in φ_1 and φ_2
   - guess equivalence relation E on V

Nelson–Oppen Method: Nondeterministic Version

input: quantifier-free conjunction φ in theory combination $T_1 \oplus T_2$

output: satisfiable or unsatisfiable

1. purification

$$\varphi \;\approx\; \varphi_1 \wedge \varphi_2 \quad \text{for } \Sigma_1\text{-formula } \varphi_1 \text{ and } \Sigma_2\text{-formula } \varphi_2$$

2. guess
   - V is set of shared variables in φ_1 and φ_2
   - guess equivalence relation E on V
   - arrangement $\alpha(V, E)$ is formula

$$\left(\bigwedge_{x\,E\,y} x = y \right) \;\wedge\; \left(\bigwedge_{\neg(x\,E\,y)} x \neq y \right)$$

Example

formula φ in combination of LIA and EUF:

$$\underbrace{1 \leqslant x \ \wedge\ x \leqslant 2 \ \wedge\ y = 1 \ \wedge\ z = 2}_{\varphi_1} \wedge \underbrace{\mathsf{f}(x) \neq \mathsf{f}(y) \wedge \mathsf{f}(x) \neq \mathsf{f}(z)}_{\varphi_2}$$

- $V = \{x, y, z\}$

Example

formula φ in combination of LIA and EUF:

$$\underbrace{1 \leqslant x \ \wedge \ x \leqslant 2 \ \wedge \ y = 1 \ \wedge \ z = 2}_{\varphi_1} \wedge \underbrace{\mathsf{f}(x) \neq \mathsf{f}(y) \wedge \mathsf{f}(x) \neq \mathsf{f}(z)}_{\varphi_2}$$

- $V = \{x, y, z\}$
- 5 different equivalence relations E:
  1. $\{\{x, y, z\}\}$
  2. $\{\{x, y\}, \{z\}\}$
  3. $\{\{x, z\}, \{y\}\}$
  4. $\{\{x\}, \{y, z\}\}$
  5. $\{\{x\}, \{y\}, \{z\}\}$

Nelson–Oppen Method: Nondeterministic Version

input: quantifier-free conjunction φ in theory combination $T_1 \oplus T_2$

output: satisfiable or unsatisfiable

1. purification

$$\varphi \;\approx\; \varphi_1 \wedge \varphi_2 \quad \text{for } \Sigma_1\text{-formula } \varphi_1 \text{ and } \Sigma_2\text{-formula } \varphi_2$$

2. guess and check
   - V is set of shared variables in φ_1 and φ_2
   - guess equivalence relation E on V
   - arrangement $\alpha(V, E)$ is formula

$$\left(\bigwedge_{x\,E\,y} x = y \right) \;\wedge\; \left(\bigwedge_{\neg(x\,E\,y)} x \neq y \right)$$

   - if $\varphi_1 \wedge \alpha(V, E)$ is T_1-satisfiable and $\varphi_2 \wedge \alpha(V, E)$ is T_2-satisfiable then return satisfiable else return unsatisfiable

Example

formula φ in combination of LIA and EUF:

$$\underbrace{1 \leqslant x \wedge x \leqslant 2 \wedge y = 1 \wedge z = 2}_{\varphi_1} \wedge \underbrace{\mathrm{f}(x) \neq \mathrm{f}(y) \wedge \mathrm{f}(x) \neq \mathrm{f}(z)}_{\varphi_2}$$

- $V = \{x, y, z\}$
- 5 different equivalence relations E:
  1. $\{\{x, y, z\}\}$ $\qquad$ $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
  2. $\{\{x, y\}, \{z\}\}$
  3. $\{\{x, z\}, \{y\}\}$
  4. $\{\{x\}, \{y, z\}\}$
  5. $\{\{x\}, \{y\}, \{z\}\}$

Example

formula φ in combination of LIA and EUF:

$$\underbrace{1 \leqslant x \;\wedge\; x \leqslant 2 \;\wedge\; y = 1 \;\wedge\; z = 2}_{\varphi_1} \;\wedge\; \underbrace{\mathsf{f}(x) \neq \mathsf{f}(y) \wedge \mathsf{f}(x) \neq \mathsf{f}(z)}_{\varphi_2}$$

- $V = \{x, y, z\}$
- 5 different equivalence relations E:

1. $\{\{x, y, z\}\}$ $\qquad$ $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
2. $\{\{x, y\}, \{z\}\}$ $\qquad$ $\varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
3. $\{\{x, z\}, \{y\}\}$
4. $\{\{x\}, \{y, z\}\}$
5. $\{\{x\}, \{y\}, \{z\}\}$

Example

formula φ in combination of LIA and EUF:

$$\underbrace{1 \leqslant x \ \wedge \ x \leqslant 2 \ \wedge \ y = 1 \ \wedge \ z = 2}_{\varphi_1} \wedge \underbrace{\mathsf{f}(x) \neq \mathsf{f}(y) \wedge \mathsf{f}(x) \neq \mathsf{f}(z)}_{\varphi_2}$$

- $V = \{x, y, z\}$
- 5 different equivalence relations E:
  1. $\{\{x, y, z\}\}$ $\qquad \varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
  2. $\{\{x, y\}, \{z\}\}$ $\qquad \varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
  3. $\{\{x, z\}, \{y\}\}$ $\qquad \varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
  4. $\{\{x\}, \{y, z\}\}$
  5. $\{\{x\}, \{y\}, \{z\}\}$

Example

formula φ in combination of LIA and EUF:

$$\underbrace{1 \leqslant x \;\wedge\; x \leqslant 2 \;\wedge\; y = 1 \;\wedge\; z = 2}_{\varphi_1} \wedge \underbrace{\mathsf{f}(x) \neq \mathsf{f}(y) \wedge \mathsf{f}(x) \neq \mathsf{f}(z)}_{\varphi_2}$$

- $V = \{x, y, z\}$
- 5 different equivalence relations E:

1. $\{\{x, y, z\}\}$ — $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
2. $\{\{x, y\}, \{z\}\}$ — $\varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
3. $\{\{x, z\}, \{y\}\}$ — $\varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
4. $\{\{x\}, \{y, z\}\}$ — $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
5. $\{\{x\}, \{y\}, \{z\}\}$

Example

formula φ in combination of LIA and EUF:

$$\underbrace{1 \leqslant x \wedge x \leqslant 2 \wedge y = 1 \wedge z = 2}_{\varphi_1} \wedge \underbrace{\mathsf{f}(x) \neq \mathsf{f}(y) \wedge \mathsf{f}(x) \neq \mathsf{f}(z)}_{\varphi_2}$$

- $V = \{x, y, z\}$
- 5 different equivalence relations E:

1. $\{\{x, y, z\}\}$ — $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
2. $\{\{x, y\}, \{z\}\}$ — $\varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
3. $\{\{x, z\}, \{y\}\}$ — $\varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
4. $\{\{x\}, \{y, z\}\}$ — $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
5. $\{\{x\}, \{y\}, \{z\}\}$ — $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable

Example

formula φ in combination of LIA and EUF:

$$\underbrace{1 \leqslant x \ \wedge\ x \leqslant 2 \ \wedge\ y = 1 \ \wedge\ z = 2}_{\varphi_1} \wedge \underbrace{\mathsf{f}(x) \neq \mathsf{f}(y) \wedge \mathsf{f}(x) \neq \mathsf{f}(z)}_{\varphi_2}$$

- $V = \{x, y, z\}$
- 5 different equivalence relations E:
  1. $\{\{x, y, z\}\}$ $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
  2. $\{\{x, y\}, \{z\}\}$ $\varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
  3. $\{\{x, z\}, \{y\}\}$ $\varphi_2 \wedge \alpha(V, E)$ is unsatisfiable
  4. $\{\{x\}, \{y, z\}\}$ $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
  5. $\{\{x\}, \{y\}, \{z\}\}$ $\varphi_1 \wedge \alpha(V, E)$ is unsatisfiable
- φ is unsatisfiable

Outline

1. **Nelson–Oppen Combination Method**

   Nondeterministic Version   Deterministic Version

2. **Quantified Boolean Formulas**

3. **PSPACE-Completeness of QBF**

4. **Further Reading**

Definition

theory T is convex if for every quantifier-free conjunctive formula F and $n \geq 1$:

- If: $F \implies \bigvee_{i=1}^{n} x_i = y_i$
- Then: $F \implies x_i = y_i$ for some $1 \leqslant i \leqslant n$

Definition

theory T is convex if for every quantifier-free conjunctive formula F and $n \geq 1$:

- If: $F \implies \bigvee_{i=1}^{n} x_i = y_i$
- Then: $F \implies x_i = y_i$ for some $1 \leqslant i \leqslant n$

Examples

- LIA is not convex

Definition

theory T is convex if for every quantifier-free conjunctive formula F and $n \geq 1$:

- If: $F \implies \bigvee_{i=1}^{n} x_i = y_i$
- Then: $F \implies x_i = y_i$ for some $1 \leqslant i \leqslant n$

Examples

- LIA is not convex

$$1 \leqslant x \leqslant 2 \ \wedge \ y = 1 \ \wedge \ z = 2 \quad \implies \quad x = y \ \vee \ x = z$$

Definition

theory T is convex if for every quantifier-free conjunctive formula F and $n \geq 1$:

- If: $F \implies \bigvee_{i=1}^{n} x_i = y_i$
- Then: $F \implies x_i = y_i$ for some $1 \leqslant i \leqslant n$

Examples

- LIA is not convex

$$1 \leqslant x \leqslant 2 \;\wedge\; y = 1 \;\wedge\; z = 2 \quad \implies \quad x = y \;\vee\; x = z$$

- LRA is convex

Definition

theory T is convex if for every quantifier-free conjunctive formula F and $n \geq 1$:

- If: $F \implies \bigvee_{i=1}^{n} x_i = y_i$
- Then: $F \implies x_i = y_i$ for some $1 \leqslant i \leqslant n$

Examples

- LIA is not convex

$$1 \leqslant x \leqslant 2 \ \wedge \ y = 1 \ \wedge \ z = 2 \quad \implies \quad x = y \ \vee \ x = z$$

- LRA is convex
- EUF is convex

Example

combination of LRA and EUF:

$$x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge\; \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \;\wedge\; z \geqslant 0$$

purification

$$\varphi_1 : \quad x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge\; w_1 = w_2 - w_3 \;\wedge\; z \geqslant 0$$

$$\varphi_2 : \quad \mathsf{f}(w_1) \neq \mathsf{f}(z) \;\wedge\; w_2 = \mathsf{f}(y) \;\wedge\; w_3 = \mathsf{f}(x)$$

Example

combination of LRA and EUF:

$$x \geqslant y \ \wedge \ y - z \geqslant x \ \wedge \ \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \ \wedge \ z \geqslant 0$$

purification

$$\varphi_1: \quad x \geqslant y \ \wedge \ y - z \geqslant x \ \wedge \ w_1 = w_2 - w_3 \ \wedge \ z \geqslant 0$$

$$\varphi_2: \quad \mathsf{f}(w_1) \neq \mathsf{f}(z) \ \wedge \ w_2 = \mathsf{f}(y) \ \wedge \ w_3 = \mathsf{f}(x)$$

implied equalities between shared variables

$$\mathcal{E}:$$

Example

combination of LRA and EUF:

$$x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge\; \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \;\wedge\; z \geqslant 0$$

purification

$$\varphi_1: \quad x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge\; w_1 = w_2 - w_3 \;\wedge\; z \geqslant 0$$

$$\varphi_2: \quad \mathsf{f}(w_1) \neq \mathsf{f}(z) \;\wedge\; w_2 = \mathsf{f}(y) \;\wedge\; w_3 = \mathsf{f}(x)$$

implied equalities between shared variables

$$\mathcal{E}:$$

test satisfiability of $\varphi_1 \;\wedge\; \mathcal{E}$

Example

combination of LRA and EUF:

$$x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge\; \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \;\wedge\; z \geqslant 0$$

purification

$$\varphi_1: \quad x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge\; w_1 = w_2 - w_3 \;\wedge\; z \geqslant 0$$

$$\varphi_2: \quad \mathsf{f}(w_1) \neq \mathsf{f}(z) \;\wedge\; w_2 = \mathsf{f}(y) \;\wedge\; w_3 = \mathsf{f}(x)$$

implied equalities between shared variables

$$\mathcal{E}: \; x = y$$

test satisfiability of $\varphi_1 \;\wedge\; \mathcal{E}$

$$\varphi_1 \;\wedge\; \mathcal{E} \implies x = y$$

Example

combination of LRA and EUF:

$$x \geqslant y \ \wedge \ y - z \geqslant x \ \wedge \ \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \ \wedge \ z \geqslant 0$$

purification

$$\varphi_1 : \quad x \geqslant y \ \wedge \ y - z \geqslant x \ \wedge \ w_1 = w_2 - w_3 \ \wedge \ z \geqslant 0$$

$$\varphi_2 : \quad \mathsf{f}(w_1) \neq \mathsf{f}(z) \ \wedge \ w_2 = \mathsf{f}(y) \ \wedge \ w_3 = \mathsf{f}(x)$$

implied equalities between shared variables

$$\mathcal{E} : \ x = y \ \wedge \ w_2 = w_3$$

test satisfiability of $\varphi_2 \ \wedge \ \mathcal{E}$

$$\varphi_2 \ \wedge \ \mathcal{E} \implies w_2 = w_3$$

Example

combination of LRA and EUF:

$$x \geqslant y \ \wedge\ y - z \geqslant x \ \wedge\ \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \ \wedge\ z \geqslant 0$$

purification

$$\varphi_1: \quad x \geqslant y \ \wedge\ y - z \geqslant x \ \wedge\ w_1 = w_2 - w_3 \ \wedge\ z \geqslant 0$$

$$\varphi_2: \quad \mathsf{f}(w_1) \neq \mathsf{f}(z) \ \wedge\ w_2 = \mathsf{f}(y) \ \wedge\ w_3 = \mathsf{f}(x)$$

implied equalities between shared variables

$$\mathcal{E}: \ x = y \ \wedge\ w_2 = w_3 \ \wedge\ z = w_1$$

test satisfiability of $\varphi_1 \ \wedge\ \mathcal{E}$

$$\varphi_1 \ \wedge\ \mathcal{E} \implies z = w_1$$

Example

combination of LRA and EUF:

$$x \geqslant y \ \wedge \ y - z \geqslant x \ \wedge \ \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \ \wedge \ z \geqslant 0$$

purification

$$\varphi_1: \quad x \geqslant y \ \wedge \ y - z \geqslant x \ \wedge \ w_1 = w_2 - w_3 \ \wedge \ z \geqslant 0$$

$$\varphi_2: \quad \mathsf{f}(w_1) \neq \mathsf{f}(z) \ \wedge \ w_2 = \mathsf{f}(y) \ \wedge \ w_3 = \mathsf{f}(x)$$

implied equalities between shared variables

$$\mathcal{E}: \ x = y \ \wedge \ w_2 = w_3 \ \wedge \ z = w_1$$

test satisfiability of $\varphi_2 \ \wedge \ \mathcal{E}$

$$\varphi_2 \ \wedge \ \mathcal{E} \ \Longrightarrow \ \bot$$

Example

combination of LRA and EUF:

$$x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge\; \mathsf{f}(\mathsf{f}(y) - \mathsf{f}(x)) \neq \mathsf{f}(z) \;\wedge\; z \geqslant 0$$

purification

$$\varphi_1: \quad x \geqslant y \;\wedge\; y - z \geqslant x \;\wedge\; w_1 = w_2 - w_3 \;\wedge\; z \geqslant 0$$

$$\varphi_2: \quad \mathsf{f}(w_1) \neq \mathsf{f}(z) \;\wedge\; w_2 = \mathsf{f}(y) \;\wedge\; w_3 = \mathsf{f}(x)$$

implied equalities between shared variables

$$\mathcal{E}: \; x = y \;\wedge\; w_2 = w_3 \;\wedge\; z = w_1$$

test satisfiability of $\varphi_2 \wedge \mathcal{E}$

$$\varphi_2 \wedge \mathcal{E} \implies \bot$$

unsatisfiable

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2
   $\mathcal{E}$: already discovered equalities between variables in V

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2
   $\mathcal{E}$: already discovered equalities between variables in V
3. test satisfiability of $\varphi_1 \wedge \mathcal{E}$

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2
   $\mathcal{E}$: already discovered equalities between variables in V
3. test satisfiability of $\varphi_1 \wedge \mathcal{E}$

   if $\varphi_1 \wedge \mathcal{E}$ is T_1-unsatisfiable then return unsatisfiable

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2
   $\mathcal{E}$: already discovered equalities between variables in V
3. test satisfiability of $\varphi_1 \wedge \mathcal{E}$

   if $\varphi_1 \wedge \mathcal{E}$ is T_1-unsatisfiable then return unsatisfiable

   else add new implied equalities between variables in V to $\mathcal{E}$

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2
   $\mathcal{E}$: already discovered equalities between variables in V
3. test satisfiability of $\varphi_1 \wedge \mathcal{E}$

   if $\varphi_1 \wedge \mathcal{E}$ is T_1-unsatisfiable then return unsatisfiable

   else add new implied equalities between variables in V to $\mathcal{E}$
4. test satisfiability of $\varphi_2 \wedge \mathcal{E}$

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2
   $\mathcal{E}$: already discovered equalities between variables in V
3. test satisfiability of $\varphi_1 \wedge \mathcal{E}$

   if $\varphi_1 \wedge \mathcal{E}$ is T_1-unsatisfiable then return unsatisfiable

   else add new implied equalities between variables in V to $\mathcal{E}$
4. test satisfiability of $\varphi_2 \wedge \mathcal{E}$

   if $\varphi_2 \wedge \mathcal{E}$ is T_2-unsatisfiable then return unsatisfiable

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2
   $\mathcal{E}$: already discovered equalities between variables in V
3. test satisfiability of $\varphi_1 \wedge \mathcal{E}$

   if $\varphi_1 \wedge \mathcal{E}$ is T_1-unsatisfiable then return unsatisfiable

   else add new implied equalities between variables in V to $\mathcal{E}$
4. test satisfiability of $\varphi_2 \wedge \mathcal{E}$

   if $\varphi_2 \wedge \mathcal{E}$ is T_2-unsatisfiable then return unsatisfiable

   else add new implied equalities between variables in V to $\mathcal{E}$

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2
   $\mathcal{E}$: already discovered equalities between variables in V
3. test satisfiability of $\varphi_1 \wedge \mathcal{E}$

   if $\varphi_1 \wedge \mathcal{E}$ is T_1-unsatisfiable then return unsatisfiable

   else add new implied equalities between variables in V to $\mathcal{E}$
4. test satisfiability of $\varphi_2 \wedge \mathcal{E}$

   if $\varphi_2 \wedge \mathcal{E}$ is T_2-unsatisfiable then return unsatisfiable

   else add new implied equalities between variables in V to $\mathcal{E}$
5. if $\mathcal{E}$ has been extended in steps 3 or 4 then goto step 2

Nelson–Oppen Method: Deterministic Version

input: quantifier-free conjunction φ in combination $T_1 \oplus T_2$ of convex theories

output: satisfiable or unsatisfiable

1. purification $\varphi \approx \varphi_1 \wedge \varphi_2$ for Σ_1-formula φ_1 and Σ_2-formula φ_2
2. V: set of shared variables in φ_1 and φ_2
   $\mathcal{E}$: already discovered equalities between variables in V
3. test satisfiability of $\varphi_1 \wedge \mathcal{E}$

   if $\varphi_1 \wedge \mathcal{E}$ is T_1-unsatisfiable then return unsatisfiable

   else add new implied equalities between variables in V to $\mathcal{E}$
4. test satisfiability of $\varphi_2 \wedge \mathcal{E}$

   if $\varphi_2 \wedge \mathcal{E}$ is T_2-unsatisfiable then return unsatisfiable

   else add new implied equalities between variables in V to $\mathcal{E}$
5. if $\mathcal{E}$ has been extended in steps 3 or 4 then goto step 2 else return satisfiable

Remark

Nelson–Oppen decision procedure can be extended to non-convex theories

Remark

Nelson–Oppen decision procedure can be extended to non-convex theories

$\Longrightarrow$ case-splitting for implied disjunction of equalities

Example

combination of LIA and EUF: $1 \leqslant x \wedge x \leqslant 2 \wedge \mathsf{f}(x) \neq \mathsf{f}(1) \wedge \mathsf{f}(x) \neq \mathsf{f}(2)$

purification

$$\varphi_1: \quad 1 \leqslant x \wedge x \leqslant 2 \wedge w_1 = 1 \wedge w_2 = 2$$

$$\varphi_2: \quad \mathsf{f}(x) \neq \mathsf{f}(w_1) \wedge \mathsf{f}(x) \neq \mathsf{f}(w_2)$$

implied equalities between shared variables

$$\mathcal{E}:$$

Remark

Nelson–Oppen decision procedure can be extended to non-convex theories

$\Longrightarrow$ case-splitting for implied disjunction of equalities

Example

combination of LIA and EUF: $1 \leqslant x \;\wedge\; x \leqslant 2 \;\wedge\; \mathsf{f}(x) \neq \mathsf{f}(1) \;\wedge\; \mathsf{f}(x) \neq \mathsf{f}(2)$

purification

$$\varphi_1: \quad 1 \leqslant x \;\wedge\; x \leqslant 2 \;\wedge\; w_1 = 1 \;\wedge\; w_2 = 2$$

$$\varphi_2: \quad \mathsf{f}(x) \neq \mathsf{f}(w_1) \;\wedge\; \mathsf{f}(x) \neq \mathsf{f}(w_2)$$

implied equalities between shared variables

$$\mathcal{E}:$$

test satisfiability of $\varphi_1 \;\wedge\; \mathcal{E}$

Remark

Nelson–Oppen decision procedure can be extended to non-convex theories

$\Longrightarrow$ case-splitting for implied disjunction of equalities

Example

combination of LIA and EUF: $1 \leqslant x \ \wedge\ x \leqslant 2 \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(1) \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(2)$

purification

$$\varphi_1: \quad 1 \leqslant x \ \wedge\ x \leqslant 2 \ \wedge\ w_1 = 1 \ \wedge\ w_2 = 2$$

$$\varphi_2: \quad \mathsf{f}(x) \neq \mathsf{f}(w_1) \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(w_2)$$

implied equalities between shared variables

$$\mathcal{E}:$$

test satisfiability of $\varphi_1 \ \wedge\ \mathcal{E} \ \Longrightarrow\ x = w_1 \ \vee\ x = w_2$

Remark

Nelson–Oppen decision procedure can be extended to non-convex theories

$\Longrightarrow$ case-splitting for implied disjunction of equalities

Example

combination of LIA and EUF: $1 \leqslant x \ \wedge \ x \leqslant 2 \ \wedge \ \mathsf{f}(x) \neq \mathsf{f}(1) \ \wedge \ \mathsf{f}(x) \neq \mathsf{f}(2)$

purification

$$\varphi_1: \quad 1 \leqslant x \ \wedge \ x \leqslant 2 \ \wedge \ w_1 = 1 \ \wedge \ w_2 = 2$$

$$\varphi_2: \quad \mathsf{f}(x) \neq \mathsf{f}(w_1) \ \wedge \ \mathsf{f}(x) \neq \mathsf{f}(w_2)$$

implied equalities between shared variables (first case)

$$\mathcal{E}: \ x = w_1$$

test satisfiability of $\varphi_1 \ \wedge \ \mathcal{E} \quad \Longrightarrow \quad x = w_1 \ \vee \ x = w_2$

case split: $x = w_1$ or $x = w_2$

Remark

Nelson–Oppen decision procedure can be extended to non-convex theories

$\Longrightarrow$ case-splitting for implied disjunction of equalities

Example

combination of LIA and EUF: $1 \leqslant x \wedge x \leqslant 2 \wedge \mathsf{f}(x) \neq \mathsf{f}(1) \wedge \mathsf{f}(x) \neq \mathsf{f}(2)$

purification

$$\varphi_1: \quad 1 \leqslant x \wedge x \leqslant 2 \wedge w_1 = 1 \wedge w_2 = 2$$

$$\varphi_2: \quad \mathsf{f}(x) \neq \mathsf{f}(w_1) \wedge \mathsf{f}(x) \neq \mathsf{f}(w_2)$$

implied equalities between shared variables (first case)

$$\mathcal{E}: \; x = w_1$$

test satisfiability of $\varphi_2 \wedge \mathcal{E} \implies \bot$

case split: $x = w_1$ or $x = w_2$

Remark

Nelson–Oppen decision procedure can be extended to non-convex theories

$\Longrightarrow$ case-splitting for implied disjunction of equalities

Example

combination of LIA and EUF: $1 \leqslant x \ \wedge\ x \leqslant 2 \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(1) \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(2)$

purification

$$\varphi_1: \quad 1 \leqslant x \ \wedge\ x \leqslant 2 \ \wedge\ w_1 = 1 \ \wedge\ w_2 = 2$$

$$\varphi_2: \quad \mathsf{f}(x) \neq \mathsf{f}(w_1) \ \wedge\ \mathsf{f}(x) \neq \mathsf{f}(w_2)$$

implied equalities between shared variables (second case)

$$\mathcal{E}: \ x = w_2$$

test satisfiability of $\varphi_2 \wedge \mathcal{E} \implies \bot$

case split: $x = w_1$ or $x = w_2$

Remark

Nelson–Oppen decision procedure can be extended to non-convex theories

$\Longrightarrow$ case-splitting for implied disjunction of equalities

Example

combination of LIA and EUF: $1 \leqslant x \ \wedge \ x \leqslant 2 \ \wedge \ \mathsf{f}(x) \neq \mathsf{f}(1) \ \wedge \ \mathsf{f}(x) \neq \mathsf{f}(2)$

purification

$$\varphi_1: \quad 1 \leqslant x \ \wedge \ x \leqslant 2 \ \wedge \ w_1 = 1 \ \wedge \ w_2 = 2$$

$$\varphi_2: \quad \mathsf{f}(x) \neq \mathsf{f}(w_1) \ \wedge \ \mathsf{f}(x) \neq \mathsf{f}(w_2)$$

implied equalities between shared variables (second case)

$$\mathcal{E}: \ x = w_2$$

test satisfiability of $\varphi_2 \ \wedge \ \mathcal{E} \quad \Longrightarrow \quad \bot$

case split: $x = w_1$ or $x = w_2$ unsatisfiable

Requirement of Convex Theories for Deterministic Nelson–Oppen

- combine LIA (non-convex) + EUF (convex)
- problem with deterministic algorithm
  - $\underbrace{1 \leqslant x \wedge x \leqslant 2 \wedge y_1 = 1 \wedge y_2 = 2}_{\varphi_{LIA}} \wedge \underbrace{f(x) = a \wedge f(y_1) \neq a \wedge f(y_2) \neq a}_{\varphi_{EUF}}$

Requirement of Convex Theories for Deterministic Nelson–Oppen

- combine LIA (non-convex) + EUF (convex)
- problem with deterministic algorithm
  - $\underbrace{1 \leqslant x \wedge x \leqslant 2 \wedge y_1 = 1 \wedge y_2 = 2}_{\varphi_{LIA}} \wedge \underbrace{f(x) = a \wedge f(y_1) \neq a \wedge f(y_2) \neq a}_{\varphi_{EUF}}$
  - no implied equalities among shared variables $\{x, y_1, y_2\}$, hence deterministic algorithm wrongly reports satisfiability

Requirement of Convex Theories for Deterministic Nelson–Oppen

- combine LIA (non-convex) + EUF (convex)
- problem with deterministic algorithm
  - $\underbrace{1 \leqslant x \wedge x \leqslant 2 \wedge y_1 = 1 \wedge y_2 = 2}_{\varphi_{LIA}} \wedge \underbrace{f(x) = a \wedge f(y_1) \neq a \wedge f(y_2) \neq a}_{\varphi_{EUF}}$
  - no implied equalities among shared variables $\{x, y_1, y_2\}$, hence deterministic algorithm wrongly reports satisfiability

Requirement of Stably Infinite Theories for Nelson–Oppen

- combine BV (not stably infinite) + EUF (stably infinite)
- problem with non-deterministic Nelson–Oppen algorithm
  - $\underbrace{x_4 \geqslant_u 0}_{\varphi_{BV}} \wedge \underbrace{\left(\bigwedge_{1 \leqslant i \leqslant 16} x_4 \neq a_i\right) \wedge \left(\bigwedge_{1 \leqslant i < j \leqslant 16} a_i \neq a_j\right)}_{\varphi_{EUF}}$

Requirement of Convex Theories for Deterministic Nelson–Oppen

- combine LIA (non-convex) + EUF (convex)
- problem with deterministic algorithm
  - $\underbrace{1 \leqslant x \wedge x \leqslant 2 \wedge y_1 = 1 \wedge y_2 = 2}_{\varphi_{LIA}} \wedge \underbrace{f(x) = a \wedge f(y_1) \neq a \wedge f(y_2) \neq a}_{\varphi_{EUF}}$
  - no implied equalities among shared variables $\{x, y_1, y_2\}$, hence deterministic algorithm wrongly reports satisfiability

Requirement of Stably Infinite Theories for Nelson–Oppen

- combine BV (not stably infinite) + EUF (stably infinite)
- problem with non-deterministic Nelson–Oppen algorithm
  - $\underbrace{x_4 \geqslant_u 0}_{\varphi_{BV}} \wedge \underbrace{\left(\bigwedge_{1 \leqslant i \leqslant 16} x_4 \neq a_i\right) \wedge \left(\bigwedge_{1 \leqslant i < j \leqslant 16} a_i \neq a_j\right)}_{\varphi_{EUF}}$
  - Nelson–Oppen reports satisfiability (only one trivial arrangement $\{x_4\}$, but the 4-bit vector x_4 cannot be different to 16 different constants $a_1, \dots, a_{16}$)

Outline

Current State

- SMT solving for various (combinations of) theories
- SMT: implicit existential quantification
- upcoming lectures: integrate quantifiers
- this lecture considers simplest case: **Q**uantified **B**oolean **F**ormulas

Definition (QBF Syntax)

$$\varphi \ ::= \ \bot \ \mid \ \top \ \mid \ x \ \mid \ \neg\varphi \ \mid \ \varphi \wedge \varphi \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \rightarrow \varphi \ \mid \ \forall x.\, \varphi \ \mid \ \exists x.\, \varphi$$

Definition (QBF Syntax)

$$\varphi ::= \bot \mid \top \mid x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \forall x.\, \varphi \mid \exists x.\, \varphi$$

Definition (QBF Semantics)

extension of propositional logic case:

$$v(\forall x.\, \varphi) = \begin{cases} \mathsf{T} & \text{if } v[x/\mathsf{F}](\varphi) = \mathsf{T} \text{ and } v[x/\mathsf{T}](\varphi) = \mathsf{T} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

Definition (QBF Syntax)

$$\varphi ::= \bot \mid \top \mid x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \forall x.\,\varphi \mid \exists x.\,\varphi$$

Definition (QBF Semantics)

extension of propositional logic case:

$$v(\forall x.\,\varphi) = \begin{cases} \mathrm{T} & \text{if } v[x/\mathrm{F}](\varphi) = \mathrm{T} \text{ and } v[x/\mathrm{T}](\varphi) = \mathrm{T} \\ \mathrm{F} & \text{otherwise} \end{cases}$$

$$v(\exists x.\,\varphi) = \begin{cases} \mathrm{T} & \text{if } v[x/\mathrm{F}](\varphi) = \mathrm{T} \text{ or } v[x/\mathrm{T}](\varphi) = \mathrm{T} \\ \mathrm{F} & \text{otherwise} \end{cases}$$

Definition (QBF Syntax)

$$\varphi \ ::= \ \bot \mid \top \mid x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \forall x.\, \varphi \mid \exists x.\, \varphi$$

Definition (QBF Semantics)

extension of propositional logic case:

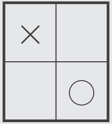
$$v(\forall x.\, \varphi) = \begin{cases} \mathsf{T} & \text{if } v[x/\mathsf{F}](\varphi) = \mathsf{T} \text{ and } v[x/\mathsf{T}](\varphi) = \mathsf{T} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

$$v(\exists x.\, \varphi) = \begin{cases} \mathsf{T} & \text{if } v[x/\mathsf{F}](\varphi) = \mathsf{T} \text{ or } v[x/\mathsf{T}](\varphi) = \mathsf{T} \\ \mathsf{F} & \text{otherwise} \end{cases}$$

Definition

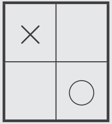
prenex normal form: $Q_1 x_1 \dots Q_n x_n.\, \varphi$ with $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and φ quantifier-free

Example (2 × 2 Tic-Tac-Toe)



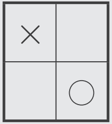
- two players (× and ○)

Example (2 $\times$ 2 Tic-Tac-Toe)



- two players ($\times$ and $\bigcirc$)
- first player ($\times$) has winning strategy

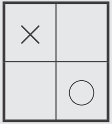
Example (2 $\times$ 2 Tic-Tac-Toe)



- two players ($\times$ and $\bigcirc$)
- first player ($\times$) has winning strategy

$$\exists \times \text{ move } \quad (\times \text{ wins or}$$

Example (2 $\times$ 2 Tic-Tac-Toe)

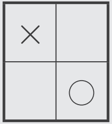


- two players ($\times$ and $\bigcirc$)
- first player ($\times$) has winning strategy

$\exists \times$ move $\quad$ ($\times$ wins or

$\forall \bigcirc$ move $\quad$ ($\bigcirc$ does not win and

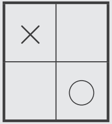
Example (2 $\times$ 2 Tic-Tac-Toe)



- two players ($\times$ and $\bigcirc$)
- first player ($\times$) has winning strategy

$$
\begin{array}{ll}
\exists \times \text{ move} & (\times \text{ wins or} \\
\forall \bigcirc \text{ move} & (\bigcirc \text{ does not win and} \\
\exists \times \text{ move} & \quad \times \text{ wins}))
\end{array}
$$

Example (2 × 2 Tic-Tac-Toe)



- two players (× and ○)
- first player (×) has winning strategy

$$\begin{array}{ll} \exists \times \text{ move} & (\times \text{ wins or} \\ \forall \bigcirc \text{ move} & (\bigcirc \text{ does not win and} \\ \exists \times \text{ move} & \quad \times \text{ wins})) \end{array}$$

- expressible in QBF

Lemma

every quantified formula can be transformed into equivalent prenex normal form

Lemma

every quantified formula can be transformed into equivalent prenex normal form

Proof

1. eliminate all propositional connectives other than $\vee$, $\wedge$, $\neg$

Lemma

every quantified formula can be transformed into equivalent prenex normal form

Proof

1. eliminate all propositional connectives other than $\vee$, $\wedge$, $\neg$
2. rename all bound variables such that every quantifier binds unique variable

Lemma

every quantified formula can be transformed into equivalent prenex normal form

Proof

1. eliminate all propositional connectives other than $\vee$, $\wedge$, $\neg$
2. rename all bound variables such that every quantifier binds unique variable
3. push propositional connectives through quantifiers:

$$
\begin{aligned}
\neg\forall x.\,\varphi &\iff \exists x.\,\neg\varphi & \neg\exists x.\,\varphi &\iff \forall x.\,\neg\varphi \\
(\forall x.\,\varphi)\wedge\psi &\iff \forall x.\,\varphi\wedge\psi & \varphi\wedge\forall x.\,\psi &\iff \forall x.\,\varphi\wedge\psi \\
(\exists x.\,\varphi)\wedge\psi &\iff \exists x.\,\varphi\wedge\psi & \varphi\wedge\exists x.\,\psi &\iff \exists x.\,\varphi\wedge\psi \\
(\forall x.\,\varphi)\vee\psi &\iff \forall x.\,\varphi\vee\psi & \varphi\vee\forall x.\,\psi &\iff \forall x.\,\varphi\vee\psi \\
(\exists x.\,\varphi)\vee\psi &\iff \exists x.\,\varphi\vee\psi & \varphi\vee\exists x.\,\psi &\iff \exists x.\,\varphi\vee\psi
\end{aligned}
$$

$$\neg\forall x.\,\varphi \iff \exists x.\,\neg\varphi \qquad \neg\exists x.\,\varphi \iff \forall x.\,\neg\varphi$$
$$(\forall x.\,\varphi)\wedge\psi \iff \forall x.\,\varphi\wedge\psi \qquad \varphi\wedge\forall x.\,\psi \iff \forall x.\,\varphi\wedge\psi$$
$$(\exists x.\,\varphi)\wedge\psi \iff \exists x.\,\varphi\wedge\psi \qquad \varphi\wedge\exists x.\,\psi \iff \exists x.\,\varphi\wedge\psi$$
$$(\forall x.\,\varphi)\vee\psi \iff \forall x.\,\varphi\vee\psi \qquad \varphi\vee\forall x.\,\psi \iff \forall x.\,\varphi\vee\psi$$
$$(\exists x.\,\varphi)\vee\psi \iff \exists x.\,\varphi\vee\psi \qquad \varphi\vee\exists x.\,\psi \iff \exists x.\,\varphi\vee\psi$$

Example

- $\neg\exists x.\,\neg((\exists y.\,(y\rightarrow x)\wedge(\neg x\vee y))\wedge\neg(\forall y.\,y\wedge x\vee\neg x\wedge\neg y))$

$$\neg\forall x.\,\varphi \iff \exists x.\,\neg\varphi \qquad \neg\exists x.\,\varphi \iff \forall x.\,\neg\varphi$$

$$(\forall x.\,\varphi)\land\psi \iff \forall x.\,\varphi\land\psi \qquad \varphi\land\forall x.\,\psi \iff \forall x.\,\varphi\land\psi$$

$$(\exists x.\,\varphi)\land\psi \iff \exists x.\,\varphi\land\psi \qquad \varphi\land\exists x.\,\psi \iff \exists x.\,\varphi\land\psi$$

$$(\forall x.\,\varphi)\lor\psi \iff \forall x.\,\varphi\lor\psi \qquad \varphi\lor\forall x.\,\psi \iff \forall x.\,\varphi\lor\psi$$

$$(\exists x.\,\varphi)\lor\psi \iff \exists x.\,\varphi\lor\psi \qquad \varphi\lor\exists x.\,\psi \iff \exists x.\,\varphi\lor\psi$$

Example

- $\neg\exists x.\,\neg((\exists y.\,(y \to x)\land(\neg x\lor y))\land\neg(\forall y.\,y\land x\lor\neg x\land\neg y))$
- $\neg\exists x.\,\neg((\exists y.\,(\neg y \lor x)\land(\neg x\lor y))\land\neg(\forall y.\,y\land x\lor\neg x\land\neg y))$

$$\neg\forall x.\,\varphi \iff \exists x.\,\neg\varphi \qquad \neg\exists x.\,\varphi \iff \forall x.\,\neg\varphi$$

$$(\forall x.\,\varphi) \wedge \psi \iff \forall x.\,\varphi \wedge \psi \qquad \varphi \wedge \forall x.\,\psi \iff \forall x.\,\varphi \wedge \psi$$

$$(\exists x.\,\varphi) \wedge \psi \iff \exists x.\,\varphi \wedge \psi \qquad \varphi \wedge \exists x.\,\psi \iff \exists x.\,\varphi \wedge \psi$$

$$(\forall x.\,\varphi) \vee \psi \iff \forall x.\,\varphi \vee \psi \qquad \varphi \vee \forall x.\,\psi \iff \forall x.\,\varphi \vee \psi$$

$$(\exists x.\,\varphi) \vee \psi \iff \exists x.\,\varphi \vee \psi \qquad \varphi \vee \exists x.\,\psi \iff \exists x.\,\varphi \vee \psi$$

Example

- $\neg\exists x.\,\neg((\exists y.\,(y \rightarrow x) \wedge (\neg x \vee y)) \wedge \neg(\forall y.\,y \wedge x \vee \neg x \wedge \neg y))$
- $\neg\exists x.\,\neg((\exists y.\,(\neg y \vee x) \wedge (\neg x \vee y)) \wedge \neg(\forall y.\,y \wedge x \vee \neg x \wedge \neg y))$
- $\neg\exists x.\,\neg((\exists y.\,(\neg y \vee x) \wedge (\neg x \vee y)) \wedge \neg(\forall z.\,z \wedge x \vee \neg x \wedge \neg z))$

$$\neg\forall x.\,\varphi \iff \exists x.\,\neg\varphi \qquad \neg\exists x.\,\varphi \iff \forall x.\,\neg\varphi$$

$$(\forall x.\,\varphi) \wedge \psi \iff \forall x.\,\varphi \wedge \psi \qquad \varphi \wedge \forall x.\,\psi \iff \forall x.\,\varphi \wedge \psi$$

$$(\exists x.\,\varphi) \wedge \psi \iff \exists x.\,\varphi \wedge \psi \qquad \varphi \wedge \exists x.\,\psi \iff \exists x.\,\varphi \wedge \psi$$

$$(\forall x.\,\varphi) \vee \psi \iff \forall x.\,\varphi \vee \psi \qquad \varphi \vee \forall x.\,\psi \iff \forall x.\,\varphi \vee \psi$$

$$(\exists x.\,\varphi) \vee \psi \iff \exists x.\,\varphi \vee \psi \qquad \varphi \vee \exists x.\,\psi \iff \exists x.\,\varphi \vee \psi$$

Example

- $\neg\exists x.\,\neg((\exists y.\,(y \rightarrow x) \wedge (\neg x \vee y)) \wedge \neg(\forall y.\,y \wedge x \vee \neg x \wedge \neg y))$
- $\neg\exists x.\,\neg((\exists y.\,(\neg y \vee x) \wedge (\neg x \vee y)) \wedge \neg(\forall y.\,y \wedge x \vee \neg x \wedge \neg y))$
- $\neg\exists x.\,\neg((\exists y.\,(\neg y \vee x) \wedge (\neg x \vee y)) \wedge \neg(\forall z.\,z \wedge x \vee \neg x \wedge \neg z))$
- $\forall x.\,(\exists y.\,(\neg y \vee x) \wedge (\neg x \vee y)) \wedge (\exists z.\,(\neg z \vee \neg x) \wedge (x \vee z))$

$$\neg\forall x.\,\varphi \iff \exists x.\,\neg\varphi \qquad \neg\exists x.\,\varphi \iff \forall x.\,\neg\varphi$$

$$(\forall x.\,\varphi) \wedge \psi \iff \forall x.\,\varphi \wedge \psi \qquad \varphi \wedge \forall x.\,\psi \iff \forall x.\,\varphi \wedge \psi$$

$$(\exists x.\,\varphi) \wedge \psi \iff \exists x.\,\varphi \wedge \psi \qquad \varphi \wedge \exists x.\,\psi \iff \exists x.\,\varphi \wedge \psi$$

$$(\forall x.\,\varphi) \vee \psi \iff \forall x.\,\varphi \vee \psi \qquad \varphi \vee \forall x.\,\psi \iff \forall x.\,\varphi \vee \psi$$

$$(\exists x.\,\varphi) \vee \psi \iff \exists x.\,\varphi \vee \psi \qquad \varphi \vee \exists x.\,\psi \iff \exists x.\,\varphi \vee \psi$$

Example

- $\neg\exists x.\,\neg((\exists y.\,(y \rightarrow x) \wedge (\neg x \vee y)) \wedge \neg(\forall y.\,y \wedge x \vee \neg x \wedge \neg y))$
- $\neg\exists x.\,\neg((\exists y.\,(\neg y \vee x) \wedge (\neg x \vee y)) \wedge \neg(\forall y.\,y \wedge x \vee \neg x \wedge \neg y))$
- $\neg\exists x.\,\neg((\exists y.\,(\neg y \vee x) \wedge (\neg x \vee y)) \wedge \neg(\forall z.\,z \wedge x \vee \neg x \wedge \neg z))$
- $\forall x.\,(\exists y.\,(\neg y \vee x) \wedge (\neg x \vee y)) \wedge (\exists z.\,(\neg z \vee \neg x) \wedge (x \vee z))$
- $\forall x.\,\exists y\,z.\,(\neg y \vee x) \wedge (\neg x \vee y) \wedge (\neg z \vee \neg x) \wedge (x \vee z)$

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$

Definition

closed QBF in prenex CNF: $Q_1 B_1 \ldots Q_n B_n . \varphi$ with quantifier-free CNF φ

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

Definition

closed QBF in prenex CNF: $Q_1 B_1 \ldots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

5 variables

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

5 variables 4 clauses

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

5 variables 4 clauses

$\exists$

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

`p cnf 5 4`	5 variables	4 clauses
`e 1 3 4 0`	$\exists\, x_1\, x_3\, x_4$	
`a 5 0`		
`e 2 0`		
`-1 2 0`		
`3 5 -2 0`		
`4 -5 -2 0`		
`-3 -4 0`		

Definition

closed QBF in prenex CNF: $Q_1 B_1 \ldots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

`p cnf 5 4`	5 variables 4 clauses
`e 1 3 4 0`	$\exists x_1 x_3 x_4$
`a 5 0`	$\forall x_5$
`e 2 0`	
`-1 2 0`	
`3 5 -2 0`	
`4 -5 -2 0`	
`-3 -4 0`	

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

`p cnf 5 4`	5 variables	4 clauses
`e 1 3 4 0`	$\exists x_1 x_3 x_4$	
`a 5 0`	$\forall x_5$	
`e 2 0`	$\exists x_2$	
`-1 2 0`		
`3 5 -2 0`		
`4 -5 -2 0`		
`-3 -4 0`		

Definition

closed QBF in prenex CNF: $Q_1 B_1 \ldots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

`p cnf 5 4`	5 variables	4 clauses
`e 1 3 4 0`	$\exists x_1 x_3 x_4$	
`a 5 0`	$\forall x_5$	
`e 2 0`	$\exists x_2$	
`-1 2 0`	$(\neg x_1 \vee x_2)$	
`3 5 -2 0`		
`4 -5 -2 0`		
`-3 -4 0`		

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

`p cnf 5 4`	5 variables 4 clauses
`e 1 3 4 0`	$\exists x_1 x_3 x_4$
`a 5 0`	$\forall x_5$
`e 2 0`	$\exists x_2$
`-1 2 0`	$(\neg x_1 \vee x_2) \wedge$
`3 5 -2 0`	$(x_3 \vee x_5 \vee \neg x_2)$
`4 -5 -2 0`	
`-3 -4 0`	

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

`p cnf 5 4`	5 variables	4 clauses
`e 1 3 4 0`	$\exists x_1 x_3 x_4$	
`a 5 0`	$\forall x_5$	
`e 2 0`	$\exists x_2$	
`-1 2 0`	$(\neg x_1 \vee x_2) \wedge$	
`3 5 -2 0`	$(x_3 \vee x_5 \vee \neg x_2) \wedge$	
`4 -5 -2 0`	$(x_4 \vee \neg x_5 \vee \neg x_2)$	
`-3 -4 0`		

Definition

closed QBF in prenex CNF: $Q_1 B_1 \dots Q_n B_n . \varphi$ with quantifier-free CNF φ QCNF

- $Q_i \in \{\forall, \exists\}$ for all $1 \leqslant i \leqslant n$ and $Q_i \neq Q_{i+1}$ for all $1 \leqslant i < n$
- B_i list of distinct variables with $B_i \cap B_j = \varnothing$ if $i \neq j$

QDIMACS Format, used in QBFLIB (QBF Satisfiability Library)

`p cnf 5 4`	5 variables 4 clauses
`e 1 3 4 0`	$\exists x_1 x_3 x_4$
`a 5 0`	$\forall x_5$
`e 2 0`	$\exists x_2$
`-1 2 0`	$(\neg x_1 \vee x_2) \wedge$
`3 5 -2 0`	$(x_3 \vee x_5 \vee \neg x_2) \wedge$
`4 -5 -2 0`	$(x_4 \vee \neg x_5 \vee \neg x_2) \wedge$
`-3 -4 0`	$(\neg x_3 \vee \neg x_4)$

Outline

1. Nelson–Oppen Combination Method
2. Quantified Boolean Formulas
3. **PSPACE-Completeness of QBF**
4. Further Reading

Definitions

- PSPACE is class of decision problems that can be solved in polynomial space by deterministic Turing machine

Definitions

- PSPACE is class of decision problems that can be solved in polynomial space by deterministic Turing machine
- decision problem A is PSPACE-hard if every PSPACE problem B is poly-time reducible to A

Definitions

- PSPACE is class of decision problems that can be solved in polynomial space by deterministic Turing machine
- decision problem A is PSPACE-hard if every PSPACE problem B is poly-time reducible to A
- decision problem A is PSPACE-complete if it is PSPACE-hard and in PSPACE

Definitions

- PSPACE is class of decision problems that can be solved in polynomial space by deterministic Turing machine
- decision problem A is PSPACE-hard if every PSPACE problem B is poly-time reducible to A
- decision problem A is PSPACE-complete if it is PSPACE-hard and in PSPACE

Theorem

- $\text{NP} \subseteq \text{PSPACE} = \text{NPSPACE}$ (open problem: $\text{NP} \subset \text{PSPACE}$)
- QBF is PSPACE-complete

Definitions

- PSPACE is class of decision problems that can be solved in polynomial space by deterministic Turing machine
- decision problem A is PSPACE-hard if every PSPACE problem B is poly-time reducible to A
- decision problem A is PSPACE-complete if it is PSPACE-hard and in PSPACE

Theorem

- NP $\subseteq$ PSPACE $=$ NPSPACE (open problem: NP $\subset$ PSPACE)
- QBF is PSPACE-complete

Some PSPACE-Complete Problems

- universality problem for regular expressions
- LTL model checking
- many games on $n \times n$ board (Othello, Go, Sokoban, Super Mario Bros.)

Lemma

QBF is in PSPACE

Lemma

QBF is in PSPACE

TQBF Algorithm

input: closed QBF formula φ in prenex normal form

output: value of φ (true or false)

Lemma

QBF is in PSPACE

TQBF Algorithm

input: closed QBF formula φ in prenex normal form

output: value of φ (true or false)

1. if φ is quantifier-free then evaluate φ and return answer

Lemma

QBF is in PSPACE

TQBF Algorithm

input: closed QBF formula φ in prenex normal form
output: value of φ (true or false)

1. if φ is quantifier-free then evaluate φ and return answer
2. if $\varphi = \exists x.\, \psi$ then return value of "TQBF($\psi[x/\bot]$) or TQBF($\psi[x/\top]$)"

Lemma

QBF is in PSPACE

TQBF Algorithm

input: closed QBF formula φ in prenex normal form
output: value of φ (true or false)

1. if φ is quantifier-free then evaluate φ and return answer
2. if $\varphi = \exists x.\, \psi$ then return value of "TQBF($\psi[x/\bot]$) or TQBF($\psi[x/\top]$)"
3. if $\varphi = \forall x.\, \psi$ then return value of "TQBF($\psi[x/\bot]$) and TQBF($\psi[x/\top]$)"

Lemma

QBF is in PSPACE

TQBF Algorithm

input: closed QBF formula φ in prenex normal form

output: value of φ (true or false)

1. if φ is quantifier-free then evaluate φ and return answer
2. if $\varphi = \exists x.\, \psi$ then return value of "TQBF($\psi[x/\bot]$) or TQBF($\psi[x/\top]$)"
3. if $\varphi = \forall x.\, \psi$ then return value of "TQBF($\psi[x/\bot]$) and TQBF($\psi[x/\top]$)"

Proof

recursive algorithm TQBF runs in linear space

Definition

deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$

Definition

deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$ with

1. Q: finite set of states

Definition

deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$ with

1. Q: finite set of states
2. Σ: finite input alphabet

Definition

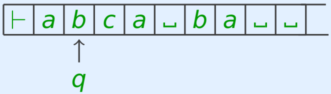
deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \textvisiblespace, \delta, s, t)$ with

1. Q: finite set of states
2. Σ: finite input alphabet
3. $\Gamma \supseteq \Sigma$: finite tape alphabet

Definition

deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \text{␣}, \delta, s, t)$ with

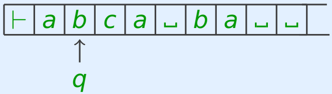
1. Q: finite set of states
2. Σ: finite input alphabet
3. $\Gamma \supseteq \Sigma$: finite tape alphabet
4. $\vdash \in \Gamma - \Sigma$: left endmarker



Definition

deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \llcorner\!\lrcorner, \delta, s, t)$ with

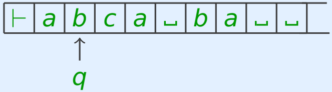
1. Q: finite set of states
2. Σ: finite input alphabet
3. $\Gamma \supseteq \Sigma$: finite tape alphabet
4. $\vdash \in \Gamma - \Sigma$: left endmarker
5. $\llcorner\!\lrcorner \in \Gamma - \Sigma$: blank symbol



Definition

deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \llcorner\!\lrcorner, \delta, s, t)$ with

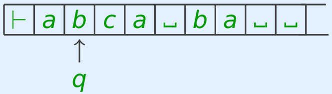
1. Q: finite set of states
2. Σ: finite input alphabet
3. $\Gamma \supseteq \Sigma$: finite tape alphabet
4. $\vdash \in \Gamma - \Sigma$: left endmarker
5. $\llcorner\!\lrcorner \in \Gamma - \Sigma$: blank symbol
6. $\delta \colon Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R\}$: (partial) transition function



Definition

deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \llcorner\!\lrcorner, \delta, s, t)$ with

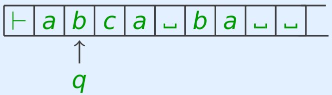
1. Q: finite set of states
2. Σ: finite input alphabet
3. $\Gamma \supseteq \Sigma$: finite tape alphabet
4. $\vdash \in \Gamma - \Sigma$: left endmarker
5. $\llcorner\!\lrcorner \in \Gamma - \Sigma$: blank symbol
6. $\delta\colon Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R\}$: (partial) transition function
7. $s \in Q$: start state



Definition

deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \textvisiblespace, \delta, s, t)$ with

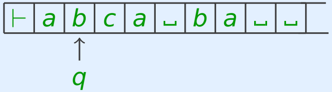
1. Q: finite set of states
2. Σ: finite input alphabet
3. $\Gamma \supseteq \Sigma$: finite tape alphabet
4. $\vdash \in \Gamma - \Sigma$: left endmarker
5. $\textvisiblespace \in \Gamma - \Sigma$: blank symbol
6. $\delta\colon Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R\}$: (partial) transition function
7. $s \in Q$: start state
8. $t \in Q$: accept state



Definition

deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$ with

1. Q: finite set of states
2. Σ: finite input alphabet
3. $\Gamma \supseteq \Sigma$: finite tape alphabet
4. $\vdash \in \Gamma - \Sigma$: left endmarker
5. $\sqcup \in \Gamma - \Sigma$: blank symbol
6. $\delta \colon Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R\}$: (partial) transition function
7. $s \in Q$: start state
8. $t \in Q$: accept state



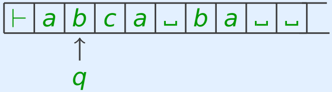
such that

$$\forall a \in \Gamma \colon \ \delta(t, a) \text{ is undefined}$$

Definition

deterministic TM (DTM) is 8-tuple $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$ with

1. Q: finite set of states
2. Σ: finite input alphabet
3. $\Gamma \supseteq \Sigma$: finite tape alphabet
4. $\vdash \in \Gamma - \Sigma$: left endmarker
5. $\sqcup \in \Gamma - \Sigma$: blank symbol
6. $\delta \colon Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R\}$: (partial) transition function
7. $s \in Q$: start state
8. $t \in Q$: accept state



such that

$$\forall a \in \Gamma\colon\ \delta(t, a) \text{ is undefined}$$

$$\forall p \in Q : \text{if } \delta(p, \vdash) \text{ is defined then } \exists q \in Q\colon\ \delta(p, \vdash) = (q, \vdash, R)$$

Definitions

- configuration: element of $Q \times \{\, y\textvisiblespace^{\omega} \mid y \in \Gamma^* \,\} \times \mathbb{N}$

Definitions

- configuration: element of $Q \times \{ y \sqcup^\omega \mid y \in \Gamma^* \} \times \mathbb{N}$
- start configuration on input $x \in \Sigma^*$: $(s, \vdash x \sqcup^\omega, 0)$

Definitions

- configuration: element of $Q \times \{ y\textvisiblespace^\omega \mid y \in \Gamma^* \} \times \mathbb{N}$
- start configuration on input $x \in \Sigma^*$: $(s, \vdash x\textvisiblespace^\omega, 0)$
- next configuration relation is binary relation $\xrightarrow[M]{1}$ defined as:

$$(p, z, n) \xrightarrow[M]{1} \begin{cases} (q, z', n-1) & \text{if } \delta(p, z_n) = (q, b, L) \\ (q, z', n+1) & \text{if } \delta(p, z_n) = (q, b, R) \end{cases}$$

Definitions

- configuration: element of $Q \times \{ y\sqcup^\omega \mid y \in \Gamma^* \} \times \mathbb{N}$
- start configuration on input $x \in \Sigma^*$: $(s, \vdash x \sqcup^\omega, 0)$
- next configuration relation is binary relation $\xrightarrow[M]{1}$ defined as:

$$(p, z, n) \xrightarrow[M]{1} \begin{cases} (q, z', n-1) & \text{if } \delta(p, z_n) = (q, b, L) \\ (q, z', n+1) & \text{if } \delta(p, z_n) = (q, b, R) \end{cases}$$

where z' is string obtained from z by substituting b for n-th symbol z_n of z

Definitions

- configuration: element of $Q \times \{y\textvisiblespace^\omega \mid y \in \Gamma^*\} \times \mathbb{N}$
- start configuration on input $x \in \Sigma^*$: $(s, \vdash x\textvisiblespace^\omega, 0)$
- next configuration relation is binary relation $\xrightarrow[M]{1}$ defined as:

$$(p, z, n) \xrightarrow[M]{1} \begin{cases} (q, z', n-1) & \text{if } \delta(p, z_n) = (q, b, L) \\ (q, z', n+1) & \text{if } \delta(p, z_n) = (q, b, R) \end{cases}$$

  where z' is string obtained from z by substituting b for n-th symbol z_n of z

- $\xrightarrow[M]{n} = (\xrightarrow[M]{1})^n \quad \forall n \geqslant 0$

Definitions

- configuration: element of $Q \times \{ y\textvisiblespace^\omega \mid y \in \Gamma^* \} \times \mathbb{N}$
- start configuration on input $x \in \Sigma^*$: $(s, \vdash x\textvisiblespace^\omega, 0)$
- next configuration relation is binary relation $\xrightarrow[M]{1}$ defined as:

$$(p, z, n) \xrightarrow[M]{1} \begin{cases} (q, z', n-1) & \text{if } \delta(p, z_n) = (q, b, L) \\ (q, z', n+1) & \text{if } \delta(p, z_n) = (q, b, R) \end{cases}$$

  where z' is string obtained from z by substituting b for n-th symbol z_n of z

- $\xrightarrow[M]{n} = (\xrightarrow[M]{1})^n \quad \forall n \geqslant 0$ $\qquad \xrightarrow[M]{*} = \bigcup_{n \geqslant 0} \xrightarrow[M]{n}$

Definitions

- configuration: element of $Q \times \{y\llcorner\!\lrcorner^\omega \mid y \in \Gamma^*\} \times \mathbb{N}$
- start configuration on input $x \in \Sigma^*$: $(s, \vdash x\llcorner\!\lrcorner^\omega, 0)$
- next configuration relation is binary relation $\xrightarrow[M]{1}$ defined as:

$$(p,z,n) \xrightarrow[M]{1} \begin{cases} (q,z',n-1) & \text{if } \delta(p,z_n) = (q,b,L) \\ (q,z',n+1) & \text{if } \delta(p,z_n) = (q,b,R) \end{cases}$$

  where z' is string obtained from z by substituting b for n-th symbol z_n of z

- $\xrightarrow[M]{n} = (\xrightarrow[M]{1})^n \quad \forall n \geqslant 0$ $\qquad \xrightarrow[M]{*} = \bigcup_{n \geqslant 0} \xrightarrow[M]{n}$
- $x \in \Sigma^*$ is accepted by M if $(s, \vdash x\llcorner\!\lrcorner^\omega, 0) \xrightarrow[M]{*} (t, y, n)$ for some y and n

Definitions

- configuration: element of $Q \times \{y\sqcup^\omega \mid y \in \Gamma^*\} \times \mathbb{N}$
- start configuration on input $x \in \Sigma^*$: $(s, \vdash x\sqcup^\omega, 0)$
- next configuration relation is binary relation $\xrightarrow[M]{1}$ defined as:

$$(p, z, n) \xrightarrow[M]{1} \begin{cases} (q, z', n-1) & \text{if } \delta(p, z_n) = (q, b, L) \\ (q, z', n+1) & \text{if } \delta(p, z_n) = (q, b, R) \end{cases}$$

  where z' is string obtained from z by substituting b for n-th symbol z_n of z

- $\xrightarrow[M]{n} = (\xrightarrow[M]{1})^n \quad \forall n \geqslant 0$ $\qquad \xrightarrow[M]{*} = \bigcup_{n \geqslant 0} \xrightarrow[M]{n}$
- $x \in \Sigma^*$ is accepted by M if $(s, \vdash x\sqcup^\omega, 0) \xrightarrow[M]{*} (t, y, n)$ for some y and n
- $L(M)$ is set of strings accepted by M

Definitions

- configuration: element of $Q \times \{ y \sqcup^\omega \mid y \in \Gamma^* \} \times \mathbb{N}$
- start configuration on input $x \in \Sigma^*$: $(s, \vdash x \sqcup^\omega, 0)$
- next configuration relation is binary relation $\xrightarrow[M]{1}$ defined as:

$$(p, z, n) \xrightarrow[M]{1} \begin{cases} (q, z', n-1) & \text{if } \delta(p, z_n) = (q, b, L) \\ (q, z', n+1) & \text{if } \delta(p, z_n) = (q, b, R) \end{cases}$$

  where z' is string obtained from z by substituting b for n-th symbol z_n of z

- $\xrightarrow[M]{n} = (\xrightarrow[M]{1})^n \quad \forall n \geqslant 0$ $\qquad \xrightarrow[M]{*} = \bigcup_{n \geqslant 0} \xrightarrow[M]{n}$
- $x \in \Sigma^*$ is accepted by M if $(s, \vdash x \sqcup^\omega, 0) \xrightarrow[M]{*} (t, \vdash \sqcup^\omega, 0)$
- $L(M)$ is set of strings accepted by M

Theorem

QBF is PSPACE-hard

Proof

- let A be arbitrary decision problem in PSPACE

Theorem

QBF is PSPACE-hard

Proof

- let A be arbitrary decision problem in PSPACE
- task: define polynomial-time reduction from A to QBF

Theorem

QBF is PSPACE-hard

Proof

- let A be arbitrary decision problem in PSPACE
- task: define polynomial-time reduction from A to QBF
- (language encoding of) A is accepted by DTM $M = (Q, \Sigma, \Gamma, \vdash, \llcorner\!\lrcorner, \delta, s, t)$ that runs in polynomial space

Theorem

QBF is PSPACE-hard

Proof

- let A be arbitrary decision problem in PSPACE
- task: define polynomial-time reduction from A to QBF
- (language encoding of) A is accepted by DTM $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$ that runs in polynomial space
- $\exists$ polynomial $p(n)$ such that M halts using at most $p(n)$ tape cells for any input x of length n

Theorem

QBF is PSPACE-hard

Proof

- let A be arbitrary decision problem in PSPACE
- task: define polynomial-time reduction from A to QBF
- (language encoding of) A is accepted by DTM $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$ that runs in polynomial space
- $\exists$ polynomial $p(n)$ such that M halts using at most $p(n)$ tape cells for any input x of length n
- given input x, we construct QBF formula $\varphi_M(x)$ such that

$$M \text{ accepts } x \quad \Longleftrightarrow \quad \varphi_M(x) \text{ is true}$$

Theorem

QBF is PSPACE-hard

Proof

- let A be arbitrary decision problem in PSPACE
- task: define polynomial-time reduction from A to QBF
- (language encoding of) A is accepted by DTM $M = (Q, \Sigma, \Gamma, \vdash, \llcorner\!\lrcorner, \delta, s, t)$ that runs in polynomial space
- $\exists$ polynomial $p(n)$ such that M halts using at most $p(n)$ tape cells for any input x of length n
- given input x, we construct QBF formula $\varphi_M(x)$ such that

$$M \text{ accepts } x \quad \Longleftrightarrow \quad \varphi_M(x) \text{ is true}$$

- continued on next slide . . .

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables
- QBF formula $\varphi_m(c_1, c_2)$ encodes that c_2 can be reached from c_1 in at most 2^m steps

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables
- QBF formula $\varphi_m(c_1, c_2)$ encodes that c_2 can be reached from c_1 in at most 2^m steps:

$$\varphi_0(c_1, c_2) \; = \; \ulcorner c_1 = c_2 \urcorner \vee \ulcorner c_1 \xrightarrow[M]{1} c_2 \urcorner$$

(cf. NP-hardness proof of SAT)

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables
- QBF formula $\varphi_m(c_1, c_2)$ encodes that c_2 can be reached from c_1 in at most 2^m steps:

$$\varphi_0(c_1, c_2) = \ulcorner c_1 = c_2 \urcorner \vee \ulcorner c_1 \xrightarrow[M]{1} c_2 \urcorner$$

(cf. NP-hardness proof of SAT)

$$\varphi_{i+1}(c_1, c_2) = \exists c.\, \varphi_i(c_1, c) \wedge \varphi_i(c, c_2)$$

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables
- QBF formula $\varphi_m(c_1, c_2)$ encodes that c_2 can be reached from c_1 in at most 2^m steps:

$$\begin{aligned} \varphi_0(c_1, c_2) &= \ulcorner c_1 = c_2 \urcorner \vee \ulcorner c_1 \xrightarrow[M]{1} c_2 \urcorner \\ \varphi_{i+1}(c_1, c_2) &= \exists c.\, \varphi_i(c_1, c) \wedge \varphi_i(c, c_2) \end{aligned}$$

(cf. NP-hardness proof of SAT)

- $\varphi_M(x) = \varphi_m(a, b)$

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables
- QBF formula $\varphi_m(c_1, c_2)$ encodes that c_2 can be reached from c_1 in at most 2^m steps:

$$\varphi_0(c_1, c_2) = \ulcorner c_1 = c_2 \urcorner \vee \ulcorner c_1 \xrightarrow[M]{1} c_2 \urcorner \qquad \text{(cf. NP-hardness proof of SAT)}$$

$$\varphi_{i+1}(c_1, c_2) = \exists c.\, \varphi_i(c_1, c) \wedge \varphi_i(c, c_2)$$

- $\varphi_M(x) = \varphi_m(a, b)$ with
  - start configuration a on input x

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables
- QBF formula $\varphi_m(c_1, c_2)$ encodes that c_2 can be reached from c_1 in at most 2^m steps:

$$\varphi_0(c_1, c_2) = \ulcorner c_1 = c_2 \urcorner \vee \ulcorner c_1 \xrightarrow[M]{1} c_2 \urcorner$$

(cf. NP-hardness proof of SAT)

$$\varphi_{i+1}(c_1, c_2) = \exists c.\, \varphi_i(c_1, c) \wedge \varphi_i(c, c_2)$$

- $\varphi_M(x) = \varphi_m(a, b)$ with
  - start configuration a on input x
  - accept configuration b

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables
- QBF formula $\varphi_m(c_1, c_2)$ encodes that c_2 can be reached from c_1 in at most 2^m steps:

$$\varphi_0(c_1, c_2) = \ulcorner c_1 = c_2 \urcorner \vee \ulcorner c_1 \xrightarrow[M]{1} c_2 \urcorner \qquad \text{(cf. NP-hardness proof of SAT)}$$

$$\varphi_{i+1}(c_1, c_2) = \exists c.\, \varphi_i(c_1, c) \wedge \varphi_i(c, c_2)$$

- $\varphi_M(x) = \varphi_m(a, b)$ with
  - start configuration a on input x
  - accept configuration b
  - bound m on required number of steps

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables
- QBF formula $\varphi_m(c_1, c_2)$ encodes that c_2 can be reached from c_1 in at most 2^m steps:

$$\varphi_0(c_1, c_2) = \ulcorner c_1 = c_2 \urcorner \vee \ulcorner c_1 \xrightarrow[M]{1} c_2 \urcorner \qquad \text{(cf. NP-hardness proof of SAT)}$$

$$\varphi_{i+1}(c_1, c_2) = \exists c.\, \varphi_i(c_1, c) \wedge \varphi_i(c, c_2)$$

- $\varphi_M(x) = \varphi_m(a, b)$ with
  - start configuration a on input x
  - accept configuration b
  - bound m on required number of steps
- $m = \mathcal{O}(p(n))$

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables
- QBF formula $\varphi_m(c_1, c_2)$ encodes that c_2 can be reached from c_1 in at most 2^m steps:

$$\varphi_0(c_1, c_2) = \ulcorner c_1 = c_2 \urcorner \vee \ulcorner c_1 \xrightarrow[M]{1} c_2 \urcorner$$

(cf. NP-hardness proof of SAT)

$$\varphi_{i+1}(c_1, c_2) = \exists c.\, \varphi_i(c_1, c) \wedge \varphi_i(c, c_2)$$

- $\varphi_M(x) = \varphi_m(a, b)$ with
  - start configuration a on input x
  - accept configuration b
  - bound m on required number of steps
- $m = \mathcal{O}(p(n))$
- size of $\varphi_M(x)$ is exponential in n ?!

Proof (cont'd)

- reachable configurations of M on input x can be encoded using $\mathcal{O}(p(n))$ variables
- QBF formula $\varphi_m(c_1, c_2)$ encodes that c_2 can be reached from c_1 in at most 2^m steps:

$$\varphi_0(c_1, c_2) = \ulcorner c_1 = c_2 \urcorner \vee \ulcorner c_1 \xrightarrow[M]{1} c_2 \urcorner \qquad \text{(cf. NP-hardness proof of SAT)}$$

$$\varphi_{i+1}(c_1, c_2) = \exists c. \forall x\, y. \ulcorner x = c_1 \urcorner \wedge \ulcorner y = c \urcorner \vee \ulcorner x = c \urcorner \wedge \ulcorner y = c_2 \urcorner \to \varphi_i(x, y)$$

- $\varphi_M(x) = \varphi_m(a, b)$ with
  - start configuration a on input x
  - accept configuration b
  - bound m on required number of steps
- $m = \mathcal{O}(p(n))$
- size of $\varphi_M(x)$ is $\mathcal{O}(p^2(n))$

Outline

1. **Nelson–Oppen Combination Method**
2. **Quantified Boolean Formulas**
3. **PSPACE-Completeness of QBF**
4. **Further Reading**

Further Reading

we will not discuss methods for QBF-solving in this course;
interested in this topic? $\longrightarrow$ look at these chapters

- Hans Kleine Büning and Uwe Bubeck
  Theory of Quantified Boolean Formulas
  Chapter 29 of Handbook of Satisfiability (second edition)
  IOS Press, 2019
- Enrico Giunchiglia, Paolo Marin, and Massimo Narizzano
  Reasoning with Quantified Boolean Formulas
  Chapter 30 of Handbook of Satisfiability (second edition)
  IOS Press, 2019

Kröning and Strichmann

- Sections 9.1, 9.2
- Chapter 10

Bradley and Manna

- Section 10.1, 10.2, 10.3

Important Concepts

- arrangement
- convex theory
- purification
- stably infinite theory
- PSPACE
- QCNF
- quantified boolean formula
- TQBF