Equality Logic and Uninterpreted Functions

Seminar: Decision Procedures

Michaela Tießler

28.06.2016

Agenda

1. Definitions
2. Use of Uninterpreted Functions
3. Decision Procedures

Equality Logic
(= Theory of Equality)

formula:	formula $\wedge$ formula | $\neg$formula | (formula) | atom
atom:	term = term
term:	identifier | constant

- Identifiers are variables defined over a single infinite domain
- Constants are elements from the same domain as the identifier

Example: $u_1 = x_1 + y_1 \wedge u_2 = x_2 + y_2 \wedge z = u_1 * u_2$

Interpretation of "=" in Equality Logic

- Reflexivity $\forall x$ $x = x$
- Symmetry $\forall x, y$ $x = y \Longrightarrow y = x$
- Transitivity $\forall x, y, z$ $x = y \ \wedge y = z \Longrightarrow x = z$

Equality Logic and Uninterpreted Functions (EUF)

formula:	formula $\wedge$ formula | $\neg$formula | (formula) | atom
atom:	term = term | predicate symbol (list of terms)
term:	identifier | function symbol (list of terms)

- Identifiers are variables defined over a single infinite domain

Example: $u_1 = F(x_1, y_1) \wedge u_2 = F(x_2, y_2) \wedge z = G(u_1, u_2)$

Interpretation of "=" in EUF

- Reflexivity $\forall x \qquad x = x$
- Symmetry $\forall x, y \qquad x = y \Longrightarrow y = x$
- Transitivity $\forall x, y, z \qquad x = y \ \wedge y = z \Longrightarrow x = z$
- Congruence $\forall t_1, \ldots, t_n, t_1', \ldots, t_n'$

$$\wedge_i \, t_i = t_i' \Longrightarrow \ F(t_1, \ldots, t_n) = F(t_1', \ldots, t_n')$$

Negation Normal Form (NNF)

- Negations are only applied to atoms
- Each formula can easily be transformed to NNF by applying three rules

$$\neg\neg A \equiv A$$
$$\neg(A \wedge B) \equiv A \vee B$$
$$\neg(A \vee B) \equiv A \wedge B$$

Agenda

1. Definitions
2. Use of Uninterpreted Functions
3. Decision Procedures

Why do we use uninterpreted functions?

- Used for abstracting, or generalizing, theorems
- Easier to reason about
- Example use: Proving Equivalence of Programs

Use Case:
Verifying a Compilation Process with Translation Validation

- Suppose that a source program contains the statement

$$z = \left(\frac{x_1}{x_2}\right)^2 * (y_1 + y_2)$$

- Outcome of the compiling process:

$$u_1 = \frac{x_1}{x_2}; \qquad u_2 = u_1 * u_1; \qquad u_3 = y_1 + y_2; \qquad z = u_2 * u_3$$

  - New auxiliary variables u_1, u_2 and u_3 have been added

- Verification condition:

$$u_1 = \frac{x_1}{x_2} \wedge u_2 = u_1 * u_1 \wedge u_3 = y_1 + y_2 \wedge z = u_2 * u_3$$

$$\Longrightarrow z = \left(\frac{x_1}{x_2}\right)^2 * (y_1 + y_2)$$

- Abstracted version:

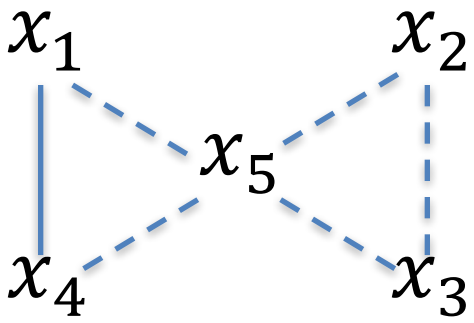
$$(u_1 = F(x_1, x_2) \wedge u_2 = G(u_1, u_1) \wedge u_3 = H(y_1, y_2) \wedge z = G(u_2, u_3)$$

$$\Longrightarrow z = G\Big(G\big(F(x_1, x_2), F(x_1, x_2)\big), H(y_1, y_2)\Big)$$

Agenda

1. Definitions
2. Use of Uninterpreted Functions
3. Decision Procedures
   - Simplify Equality Formula
   - Ackermann`s Reduction
   - Congruence Closure

Simplify Equality Formula
Equality Graphs



- Dashed edges represent equalities
- Solid edges represent disequalities

Simplify Equality Formula Algorithm

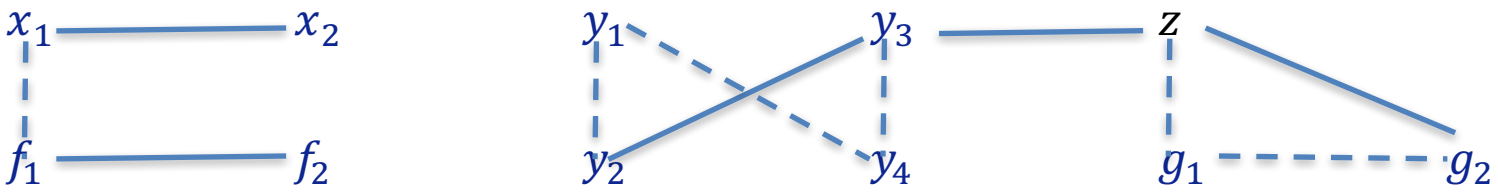
- Input: Equality formula φ^E

1. Let $\varphi^{E\prime} = \varphi^E$
2. Construct the equality graph $\mathrm{G}^E(\varphi^{E\prime})$
3. Replace each pure literal "x = y" or "x ≠ y" in $\varphi^{E\prime}$ with TRUE, if (x, y) not in a simple contradictory cycle
4. If anything changed in the last two steps go to step 2
5. Return $\varphi^{E\prime}$

Simplify Equality Formula Example

$$\varphi^E := \begin{pmatrix} (\mathrm{x}_1 \neq x_2 \vee f_1 \neq \mathrm{f}_2 \vee \mathrm{x}_1 = \mathrm{f}_1) \wedge \\ (y_1 = y_2 \vee y_3 = y_4) \wedge y_1 = y_4 \wedge y_2 \neq y_4 \\ (z = g_1 \wedge g_1 = g_2 \wedge z \neq g_2) \wedge z \neq y_3 \end{pmatrix}$$

- Equality Graph



Simplify Equality Formula Example

$$\varphi^{E} := \begin{pmatrix} (\mathrm{x}_1 \neq x_2 \vee f_1 \neq \mathrm{f}_2 \vee \mathrm{x}_1 = \mathrm{f}_1) \wedge \\ (y_1 = y_2 \vee y_3 = y_4) \wedge y_1 = y_4 \wedge y_2 \neq y_4 \\ (z = g_1 \wedge g_1 = g_2 \wedge z \neq g_2) \wedge z \neq y_3 \end{pmatrix}$$

- Replacing literals

$$\varphi^{E\prime} := \begin{pmatrix} (TRUE \vee TRUE \vee TRUE) \wedge \\ (y_1 = y_2 \vee y_3 = y_4) \wedge y_1 = y_4 \wedge y_2 \neq y_4 \\ (z = g_1 \wedge g_1 = g_2 \wedge z \neq g_2) \wedge TRUE \end{pmatrix}$$

Simplify Equality Formula Example

$$\varphi^E := \begin{pmatrix} (\mathrm{x}_1 \neq x_2 \vee f_1 \neq \mathrm{f}_2 \vee \mathrm{x}_1 = \mathrm{f}_1) \wedge \\ (y_1 = y_2 \vee y_3 = y_4) \wedge y_1 = y_4 \wedge y_2 \neq y_4 \\ (z = g_1 \wedge g_1 = g_2 \wedge z \neq g_2) \wedge z \neq y_3 \end{pmatrix}$$

- Simplification

$$\varphi^{E\prime} := \begin{pmatrix} (y_1 = y_2 \vee y_3 = y_4) \wedge y_1 = y_4 \wedge y_2 \neq y_4 \\ z = g_1 \wedge g_1 = g_2 \wedge z \neq g_2 \end{pmatrix}$$

Simplify Equality Formula Example

$$\varphi^{E\prime} := \begin{pmatrix} (y_1 = y_2 \vee y_3 = y_4) \wedge y_1 = y_4 \wedge y_2 \neq y_4 \\ z = g_1 \wedge g_1 = g_2 \wedge z \neq g_2 \end{pmatrix}$$

- Reconstructing Equality Graph

→ no further changes possible

Ackermann`s Reduction

- Transforms EUF formula φ^{UF} to an equality logic formula φ^{E} of the form:

$$\varphi^{E} := FC^{E} \Longrightarrow flat^{E}$$

  Where FC^{E} is a conjunction of functional consistency constraints and $flat^{E}$ is a flattening of φ^{UF}
- Functional consistency: Instances of the same function return the same value if given equal arguments
- φ^{E} is valid if and only if φ^{UF} is valid

Ackermann`s Reduction Algorithm

1. Indexing the functions, beginning with the inner functions
2. Take the original expression φ^{UF} and transform it to its flat $flat^{E}$
3. Compute conjunction of constraints FC^{E}
4. Let $\varphi^{E} := FC^{E} \Longrightarrow flat^{E}$

Ackermann`s Reduction Example

$$x_1 = x_2 \wedge y_1 = y_2 \Longrightarrow \underbrace{F\big(\overbrace{G\big(\underbrace{G(x_1)}_{g_1}\big)}^{g_2}, y_1\big)}_{f_1} = \underbrace{F\big(\overbrace{G\big(\underbrace{G(x_2)}_{g_3}\big)}^{g_4}, y_2\big)}_{f_2}$$

1. Indexing the functions, beginning with the inner functions

$$g_1 := G(x_1) \qquad g_3 := G(x_2)$$
$$g_2 := G(g_1) \qquad g_4 := G(g_3)$$
$$f_1 := F(g_2) \qquad f_2 := F(g_4)$$

Ackermann`s Reduction Example

$$x_1 = x_2 \wedge y_1 = y_2 \Longrightarrow \underbrace{F\big(\overbrace{G\big(\underbrace{G(x_1)}_{g_1}\big)}^{g_2}, y_1\big)}_{f_1} = \underbrace{F\big(\overbrace{G\big(\underbrace{G(x_2)}_{g_3}\big)}^{g_4}, y_2\big)}_{f_2}$$

2. Take the original expression φ^{UF} and transform it to its flat $flat^E$

$$flat_E := x_1 = x_2 \wedge y_1 = y_2 \Longrightarrow f_1 = f_2$$

Ackermann`s Reduction Example

$$x_1 = x_2 \wedge y_1 = y_2 \Longrightarrow \underbrace{F\big(\overbrace{G\big(\underbrace{G(x_1)}_{g_1}\big)}^{g_2}, y_1\big)}_{f_1} = \underbrace{F\big(\overbrace{G\big(\underbrace{G(x_2)}_{g_3}\big)}^{g_4}, y_2\big)}_{f_2}$$

3. Compute conjunction of constraints

$$FC^E := \quad x_1 = x_2 \Longrightarrow g_1 = g_3 \quad \wedge \quad g_1 = g_3 \Longrightarrow g_2 = g_4 \quad \wedge$$

$$g_2 = g_4 \wedge y_1 = y_2 \Longrightarrow f_1 = f_2$$

$$x_1 = g_1 \Longrightarrow g_1 = g_2 \quad \wedge \quad x_2 = g_3 \Longrightarrow g_3 = g_4 \quad \wedge$$

$$g_1 = g_2 \Leftrightarrow g_3 = g_4$$

Ackermann`s Reduction Example

$$x_1 = x_2 \wedge y_1 = y_2 \Longrightarrow \underbrace{F\big(\overbrace{G\big(\underbrace{G(x_1)}_{g_1}\big)}^{g_2}, y_1\big)}_{f_1} = \underbrace{F\big(\overbrace{G\big(\underbrace{G(x_2)}_{g_3}\big)}^{g_4}, y_2\big)}_{f_2}$$

4. Let $\varphi^E := FC^E \Longrightarrow flat^E$

$$\varphi^E := \quad (x_1 = x_2 \Longrightarrow g_1 = g_3 \quad \wedge \quad g_1 = g_3 \Longrightarrow g_2 = g_4 \quad \wedge$$
$$g_2 = g_4 \wedge y_1 = y_2 \Longrightarrow f_1 = f_2 \quad \wedge$$
$$x_1 = g_1 \Longrightarrow g_1 = g_2 \quad \wedge \quad x_2 = g_3 \Longrightarrow g_3 = g_4 \quad \wedge$$
$$g_1 = g_2 \Leftrightarrow g_3 = g_4) \quad \Longrightarrow$$
$$(x_1 = x_2 \wedge y_1 = y_2 \Longrightarrow f_1 = f_2)$$

Congruence Closure

- Algorithm to decide a conjunction of equalities and uninterpreted functions
- Input: A conjunction φ^{UF} of equality predicates over variables and uninterpreted functions in negation normal form
- Output:
  - equivalence classes
  - Satisfiability of φ^{UF}

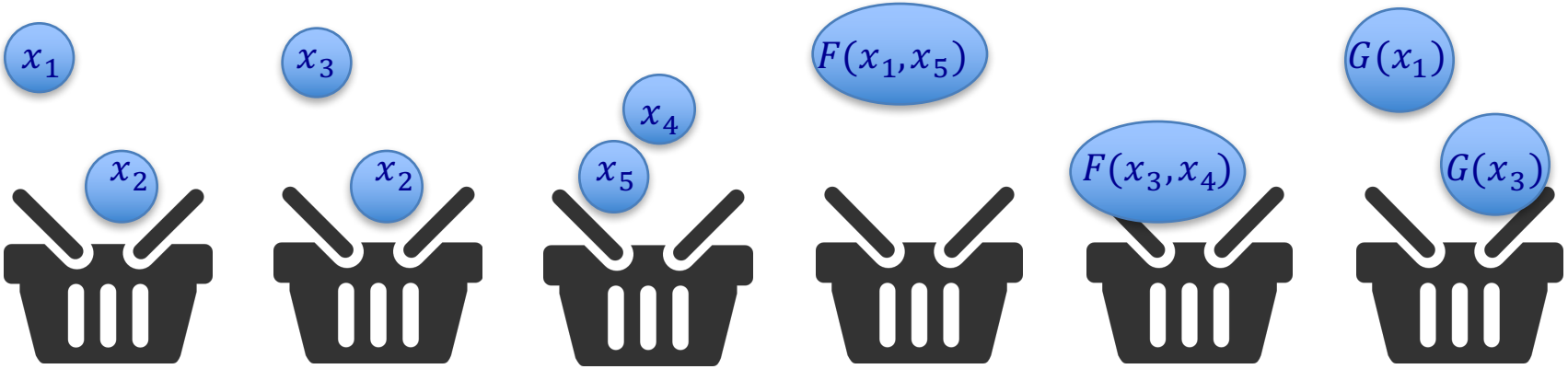
Congruence Closure Algorithm

1. Put two terms t_1, t_2 in their own equivalence class if $(t_1 = t_2)$ is a predicate in φ^{UF}
2. Merge equivalence classes with a shared term until there are no more classes to be merged
3. Merge terms $\mathrm{F}(t_i)$ and $\mathrm{F}(t_j)$ if t_i and t_j are in the same equivalence class
4. φ^{UF} *is „Satisfiable“ if there doesn‘t exist any unequality* $t_i \neq t_j$ with t_i and t_j of the same equivalence class

Congruence Closure Example

$$\varphi^{UF} := x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1 \wedge F(x_1, x_5) \neq F(x_3, x_4) \wedge G(x_1) = G(x_3)$$

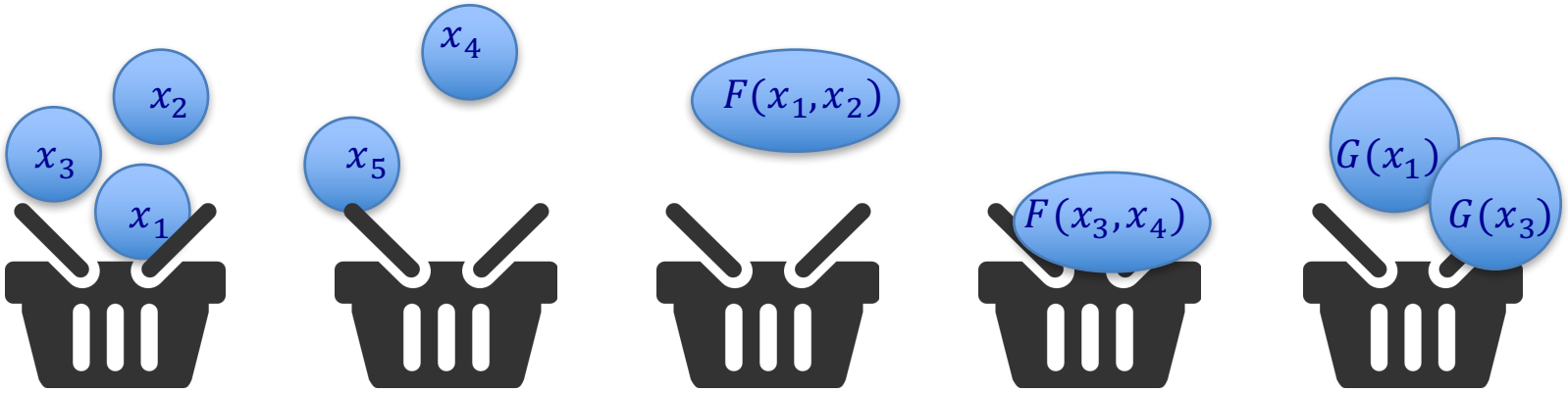
1. Put two terms t_1, t_2 in their own equivalence class if $(t_1 = t_2)$ is a predicate in φ^{UF}



Congruence Closure Example

$$\varphi^{UF} := x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1 \wedge F(x_1, x_5) \neq F(x_3, x_4) \wedge G(x_1) = G(x_3)$$

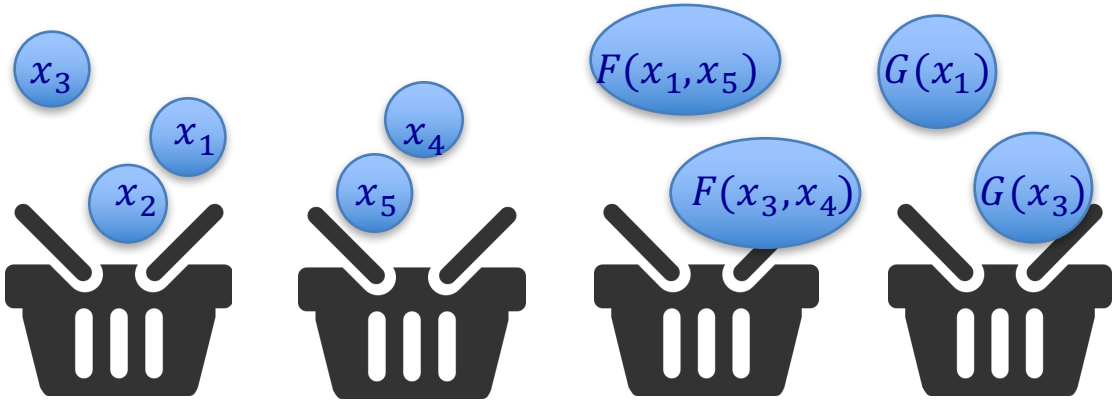
2. Merge equivalence classes with a shared term until there are no more classes to be merged



Congruence Closure Example

$$\varphi^{UF} := x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1 \wedge F(x_1, x_5) \neq F(x_3, x_4) \wedge G(x_1) = G(x_3)$$

3. Merge terms $\mathrm{F}(t_i)$ and $\mathrm{F}(t_j)$ if t_i and t_j are in the same equivalence class



Congruence Closure Example

$$\varphi^{UF} := x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1 \wedge F(x_1, x_5) \neq F(x_3, x_4) \wedge G(x_1) = G(x_3)$$

4. *φ^{UF} is „Satisfiable“ if there doesn‘t exist any unequality $t_i \neq t_j$* with t_i and t_j of the same equivalence class

$$\{x_1, x_2, x_3\}, \{x_4, x_5\},$$
$$\{F(x_1, x_2), F(x_3, x_4)\}, \{G(x_1)\}, \{G(x_5)\}$$
$$F(x_1, x_5) \neq F(x_3, x_4) \Longrightarrow Unsatisfiable$$

Congruence Closure based on Graphs

$G = (V, E)$	directed Graph	$\lambda(v)$	label
u, v	vertices	R	Relation on V

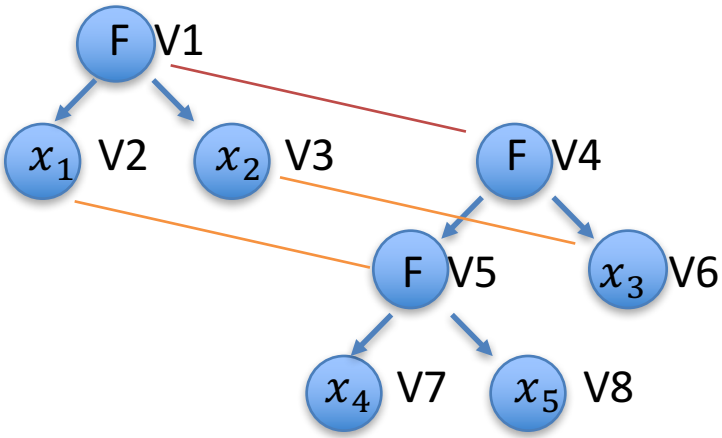
- u, v congruent under R if

$$\lambda(u) = \lambda(v) \;\&\&\; \forall u_i, v_i \text{ with } V(u) = \{u_1, \dots u_n\},$$
$$V(v) = \{v_1, \dots v_n\}\, \{u(i), v(i)\} \epsilon\, R \Longrightarrow \{u, v\} \in R$$

- u, v closed under congruences R if

$$\forall u, v \text{ with } u, v \text{ congruent under } R$$
$$\exists\, R' \text{ minimal extention of R;}$$
$$\text{with } R' \text{equivalence Relation } \&\&\; R' \text{ closed under } R$$

Congruence Closure based on Graphs Example



- R Relation $\{V1, V4\}$
- V1, V4 congruent under R
  - ➢ Congruence closure of R must include at least $\{V2, V5\}$ and $\{V3, V6\}$
- ➢ Minimal equivalence Relation:

$$\{\{V1, V4\}, \{V2, V5\}, \{V3, V6\}\}$$

Congruence Closure
Example for Implementation

- Based on the operations *union* and *find*:
  - *find(u)*: returns the unique name of the equivalence class *u* belongs to
  - *union(u, v)*: combines equivalence classes of vertices *u* and *v*
- Other operations:
  - *congruent(u, v)*: returns true if *u* and *v* are congruent

Congruence Closure
Example for Implementation

Compute congruence closure

merge (u, v)
    if find(u)=find(v) then return
    P_u = set of predecessors of all vertices equivalent to u
    P_v = set of predecessors of all vertices equivalent to v
    union(u, v)
    For each pair (x, y) with $x \in P_u$ and $y \in P_v$
        if find (u) $\neq$ find (v) && congruent(u,v) = TRUE then merge(x, y)

Congruence Closure
Example for Implementation

Decision Procedure

$$t_1 = u_1 \wedge \cdots \wedge t_p = u_p \wedge r_1 = s_1 \wedge \cdots \wedge r_q = s_1$$

Construct Graph G which corresponds to the set of all terms appearing in the conjunction

let $\tau(t)$ be the vertex in G representing t

let R be the identity relation on the vertices of G

for $1 \leq i \leq p$ merge $\left(\tau(t_i), \tau(u_i)\right)$

for $1 \leq i \leq q$

if $\tau(\mathrm{r_i})$, $\tau(s_i)$ equivalent then return UNSATISFIABLE

return SATISFIABLE

Congruence Closure Complexity

- Graph based CC: $\mathcal{O}(nm)$

  with n number of vertices of G and m number of edges of G

- Average time of fastest CC algorithm:

$$\mathcal{O}(n \log n)$$

  with n length of conjunction

Further approaches

- Graph-Based Reduction to Propositional Logic
- Solving equality logic formulas by relying on the small-model property of this logic