Formale Methoden der Informatik

Block 2: Satisfiability Problems

4. A Decision Procedure for Equality Logic

Uwe Egly

Knowledge-Based Systems Group
Institute of Information Systems
Vienna University of Technology

The overall procedure again

Goal of the SAT part

Provide necessary tools and background info to construct a decision procedure for equality logic with uninterpreted functions (EUF).

Overall procedure

The problem of deciding a EUF-formula φ^{EUF} is reduced to the SAT problem of a propositional formula φ^{P} such that

$$\varphi^{EUF} \text{ is } E\text{-valid} \iff \varphi^{E} \text{ is } E\text{-valid} \iff \varphi^{P} \text{ is unsat}$$

Then a model of φ^{P} provides a counterexample to the E-validity of φ^{EUF}!

The task for today

- Given a formula φ^E (this can be the negation of some ψ^E). We want to construct a propositional formula φ^P such that

$$\boxed{\varphi^E \text{ is } E\text{-satisfiable} \quad \text{iff} \quad \varphi^P \text{ is satisfiable}}$$

- The construction of φ^P from φ^E (reduction) requires run time polynomial in the size of φ^E.
- The reduction uses several kinds of (equality) graphs.
- An overview of the translation can be found in:
  D. Kroening, O. Strichman. Decision Procedures, Springer, 2008

Outline

Equality logic

- We work in a restricted quantifier-free fragment of $\mathcal{T}_E$.
- Σ consists only of constants, Boolean variables and $\doteq$.
- That is, we have no FSs and other PSs of arity > 0.

Why equality logic?

- Technically not necessary
  (E-logic has the same expressive power as propositional logic)
- It can be shown that the satisfiability problem for equality logic is NP-complete.
- But often allow for more natural problem descriptions especially if E-logic is extended by uninterpreted functions.

☞ Examples and applications will be discussed in the next lecture.

The syntax of equality logic

Definition (The syntax of equality logic (E-logic))

$$
\begin{array}{rcl}
\mathit{formula} & ::= & \mathit{atom} \mid (\mathit{formula}) \mid \neg \mathit{formula} \mid \mathit{formula} \wedge \mathit{formula} \mid \\
 & & \mathit{formula} \vee \mathit{formula} \mid \mathit{formula} \rightarrow \mathit{formula} \\
\mathit{atom} & ::= & \mathit{term} \doteq \mathit{term} \mid \mathit{Boolean\ variable} \\
\mathit{term} & ::= & \mathit{identifier} \mid \mathit{constant}
\end{array}
$$

Remarks

1. The set of Boolean vars and the set of term vars are disjoint!
2. $\doteq$ is always interpreted as equality!
3. The connectives $\leftrightarrow$ and $\oplus$ are not part of the language.
4. An E-formula is a formula of E-logic.

Example

Consider formula $\varphi\colon x_1 \doteq x_2 \wedge (x_2 \doteq x_3 \vee (\neg(x_1 \doteq x_3) \wedge x_1 \doteq 2))$

- Let $\Sigma = (\{1/0, 2/0, 3/0\}, \{\doteq/2\})$
- Let $\mathcal{U} = \mathbb{N}$, interpret any constant by itself and assign natural numbers to variables as follows:

$$\{x_1 \mapsto 2,\ x_2 \mapsto 2,\ x_3 \mapsto 9\}$$

- Then φ is true in this structure.

Removal of constants from E-formulas

Definition (Removal of constants from an E-formula)

Let φ^E be any E-formula with constants $c_1, \ldots, c_n$. Construct an E-formula ψ^E without any constant by replacing each c_i by v_{c_i}, where v_{c_i} is a new term variable (identifier).

Observations

1. The length of φ^E is equal to the length of ψ^E.
2. φ^E and ψ^E are equi-satisfiable.
   ☞ See extra-sheet4-1.pdf in TUWEL for a proof sketch!

Removal of Boolean variables from E-formulas

Definition (Removal of Boolean variables from an E-formula)

Let φ^E be any E-formula with Boolean variables $b_1, \ldots, b_n$. Construct an E-formula ψ^E without any Boolean variable by replacing each b_i by $v_{b_i,1} \doteq v_{b_i,2}$, where $v_{b_i,1}$, $v_{b_i,2}$ are two new term variables (identifiers).

Observations

1. The number of atoms in φ^E and ψ^E are identical.
2. φ^E and ψ^E are equi-satisfiable.
   ☞ See extra-sheet4-1a.pdf in TUWEL for a proof!

Equality and disequality literals

- We consider E-formulas φ^E
  - in negation normal form (NNF) and
  - without constants and Boolean variables.
- The set of all atoms in φ^E is denoted by $At(\varphi^E)$.

Definition

The set of equality literals of φ^E, $E_{\doteq}$, is the set of positive literals (equalities) in φ^E. Likewise for the set of disequality literals of φ^E, $E_{\dot{\neq}}$, and negative literals (disequalities).

Example: $\varphi^E : x \doteq y \wedge y \doteq z \wedge z \dot{\neq} x$

$$E_{\doteq} = \{x \doteq y,\ y \doteq z\} \qquad E_{\dot{\neq}} = \{z \dot{\neq} x\}$$

Equality graphs

Definition

The equality graph, $G^E(\varphi^E)$, for an E-formula φ^E in NNF and without constants and Boolean variables is an undirected graph $(V, E_{\doteq}, E_{\dot{\neq}})$, where the nodes (or vertices) V corresponds to the variables in φ^E and the two edge sets correspond to the set of equality literals and disequality literals, respectively.

NB: $E_{\doteq}$ and $E_{\dot{\neq}}$ represent literal sets and edges in the graph.

Example: $G^E(\varphi^E)$ with $\varphi^E : x \doteq y \wedge y \doteq z \wedge z \dot{\neq} x$

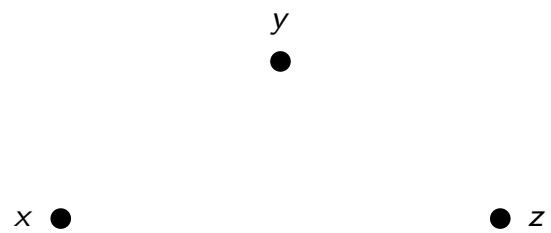
Equality graphs

Definition

The equality graph, $G^E(\varphi^E)$, for an E-formula φ^E in NNF and without constants and Boolean variables is an undirected graph $(V, E_{\doteq}, E_{\dot{\neq}})$, where the nodes (or vertices) V corresponds to the variables in φ^E and the two edge sets correspond to the set of equality literals and disequality literals, respectively.

NB: $E_{\doteq}$ and $E_{\dot{\neq}}$ represent literal sets and edges in the graph.

Example: $G^E(\varphi^E)$ with $\varphi^E : x \doteq y \wedge y \doteq z \wedge z \dot{\neq} x$



The variables of φ^E are the vertices of $G^E(\varphi^E)$

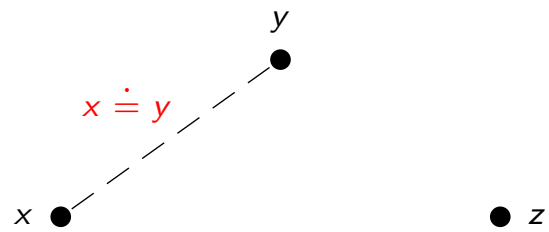
Equality graphs

Definition

The equality graph, $G^E(\varphi^E)$, for an E-formula φ^E in NNF and without constants and Boolean variables is an undirected graph $(V, E_{\doteq}, E_{\dot{\neq}})$, where the nodes (or vertices) V corresponds to the variables in φ^E and the two edge sets correspond to the set of equality literals and disequality literals, respectively.

NB: $E_{\doteq}$ and $E_{\dot{\neq}}$ represent literal sets and edges in the graph.

Example: $G^E(\varphi^E)$ with $\varphi^E : x \doteq y \wedge y \doteq z \wedge z \dot{\neq} x$



Adding an equality edge . . .

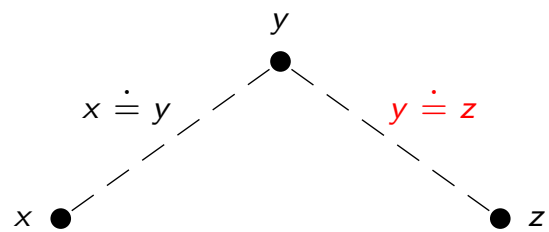
Equality graphs

Definition

The equality graph, $G^E(\varphi^E)$, for an E-formula φ^E in NNF and without constants and Boolean variables is an undirected graph $(V, E_{\doteq}, E_{\dot{\neq}})$, where the nodes (or vertices) V corresponds to the variables in φ^E and the two edge sets correspond to the set of equality literals and disequality literals, respectively.

NB: $E_{\doteq}$ and $E_{\dot{\neq}}$ represent literal sets and edges in the graph.

Example: $G^E(\varphi^E)$ with $\varphi^E : x \doteq y \wedge y \doteq z \wedge z \dot{\neq} x$



. . . and another one . . .

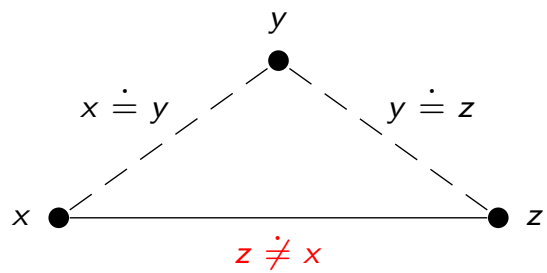
Equality graphs

Definition

The equality graph, $G^E(\varphi^E)$, for an E-formula φ^E in NNF and without constants and Boolean variables is an undirected graph $(V, E_{\doteq}, E_{\dot{\neq}})$, where the nodes (or vertices) V corresponds to the variables in φ^E and the two edge sets correspond to the set of equality literals and disequality literals, respectively.

NB: $E_{\doteq}$ and $E_{\dot{\neq}}$ represent literal sets and edges in the graph.

Example: $G^E(\varphi^E)$ with $\varphi^E : x \doteq y \wedge y \doteq z \wedge z \dot{\neq} x$



. . . and finally a disequality edge

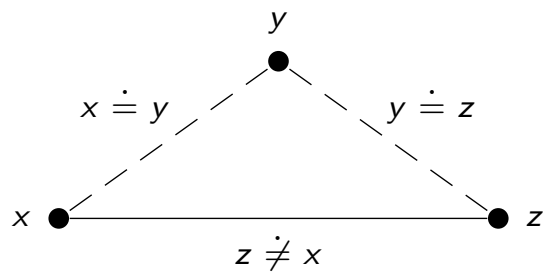
Equality graphs

Definition

The equality graph, $G^E(\varphi^E)$, for an E-formula φ^E in NNF and without constants and Boolean variables is an undirected graph $(V, E_{\doteq}, E_{\dot{\neq}})$, where the nodes (or vertices) V corresponds to the variables in φ^E and the two edge sets correspond to the set of equality literals and disequality literals, respectively.

NB: $E_{\doteq}$ and $E_{\dot{\neq}}$ represent literal sets and edges in the graph.

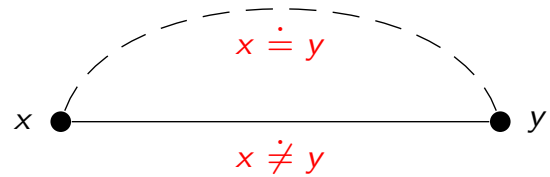
Example: $G^E(\varphi^E)$ with $\varphi^E : x \doteq y \wedge y \doteq z \wedge z \dot{\neq} x$



Observation: $G^E(\varphi^E)$ is independent from φ^E's $(\wedge, \vee)$-structure!

Equality graphs and loss of information

Consider the following equality graph $G^E(\varphi^E)$:



What is the formula φ^E?

Possibility 1: φ^E is $x \doteq y \vee x \neq y$ a tautology

Possibility 2: φ^E is $x \doteq y \wedge x \neq y$ a contradiction

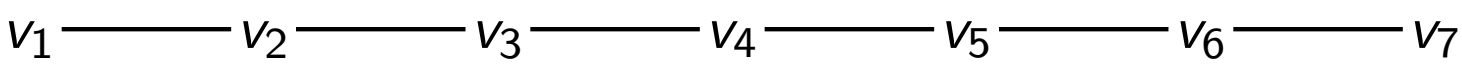
- ☞ The same graph represents a tautology and a contradiction.
- ☞ We loose the information about the propositional structure when we consider $G^E(\varphi^E)$ instead of φ^E!

Paths and cycles

- A path in a graph $G = (V, E)$ is a sequence $v_1, \ldots, v_n$ of vertices such that, for all $1 \leq i < n$, $(v_i, v_{i+1}) \in E$.
- v_1 is the start vertex and v_n is the end vertex.
- We often unify a path with its associated edges.
- The length of a path $v_1, \ldots, v_n$ is the number of its edges.
- A simple path in G is a path without repeated vertices.
- A cycle in $G = (V, E)$ is a path $v_1, \ldots, v_{n-1}, v_1$ with repeated vertices.
- A simple cycle in G is a cycle where v_1 $(=v_n)$ is the only repeated vertex.

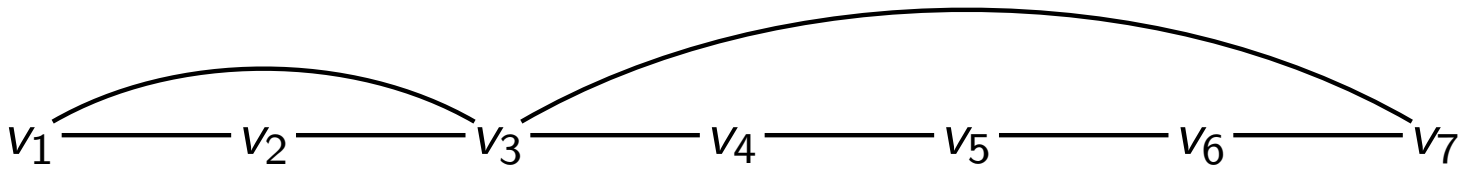
Examples of paths and cycles

A simple path $v_1, v_2, v_3, v_4, v_5, v_6, v_7$ of length 6.



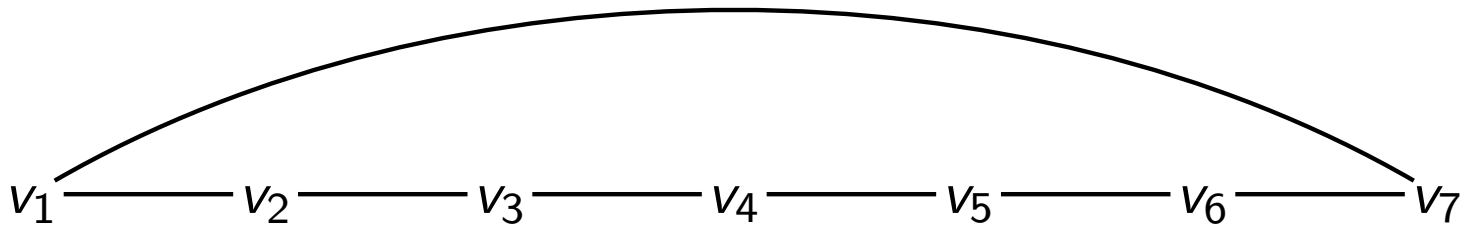
Examples of paths and cycles

A cycle $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_3, v_1$.



Examples of paths and cycles

A simple cycle $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_1$.



Equality and disequality paths

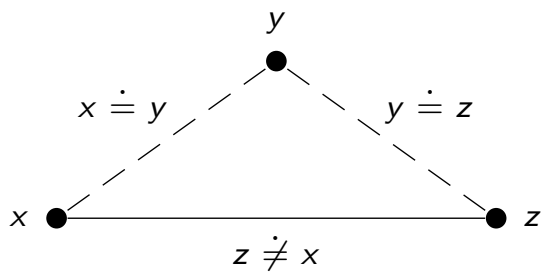
Definition (equality path, disequality path, simple paths)

An equality (eq) path in $G^E(\varphi^E)$ is a path consisting of edges from $E_{\dot{=}}$. The existence of an equality path between vertices x and y is denoted by $x \dot{=}^* y$.

A disequality (diseq) path in $G^E(\varphi^E)$ is a path consisting of edges from $E_{\dot{=}}$ and exactly one edge from $E_{\dot{\neq}}$. The existence of a disequality path between vertices x and y is denoted by $x \dot{\neq}^* y$.

A simple (eq or diseq) path in $G^E(\varphi^E)$ is a path without cycles.

Example: Reconsider $G^E(\varphi^E)$



Equality and disequality paths

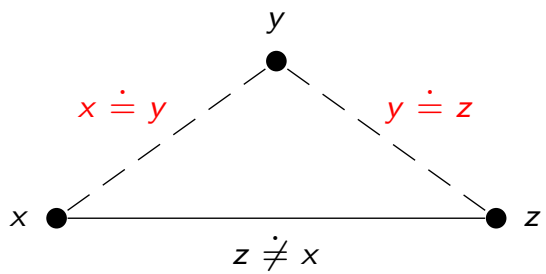
Definition (equality path, disequality path, simple paths)

An equality (eq) path in $G^E(\varphi^E)$ is a path consisting of edges from $E_{\doteq}$. The existence of an equality path between vertices x and y is denoted by $x \doteq^* y$.

A disequality (diseq) path in $G^E(\varphi^E)$ is a path consisting of edges from $E_{\doteq}$ and exactly one edge from $E_{\dot{\neq}}$. The existence of a disequality path between vertices x and y is denoted by $x \dot{\neq}^* y$.

A simple (eq or diseq) path in $G^E(\varphi^E)$ is a path without cycles.

Example: Reconsider $G^E(\varphi^E)$



$x \doteq^* z$: $x \doteq y$, $y \doteq z$

Equality and disequality paths

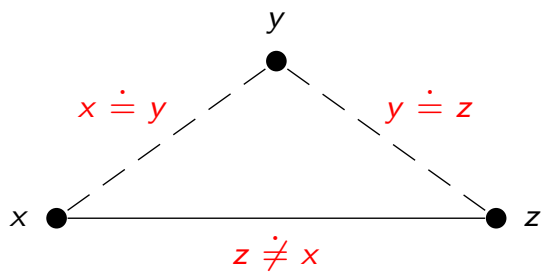
Definition (equality path, disequality path, simple paths)

An equality (eq) path in $G^E(\varphi^E)$ is a path consisting of edges from $E_{\doteq}$. The existence of an equality path between vertices x and y is denoted by $x \doteq^* y$.

A disequality (diseq) path in $G^E(\varphi^E)$ is a path consisting of edges from $E_{\doteq}$ and exactly one edge from $E_{\dot{\neq}}$. The existence of a disequality path between vertices x and y is denoted by $x \dot{\neq}^* y$.

A simple (eq or diseq) path in $G^E(\varphi^E)$ is a path without cycles.

Example: Reconsider $G^E(\varphi^E)$



$x \doteq^* z$: $x \doteq y$, $y \doteq z$

$x \dot{\neq}^* x$: $x \doteq y$, $y \doteq z$, $z \dot{\neq} x$

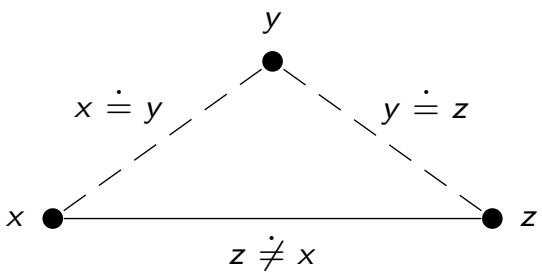
Contradictory cycles

Definition

A contradictory cycle (CC) in $G^E(\varphi^E)$ is a cycle with exactly one disequality edge.

Important Property:

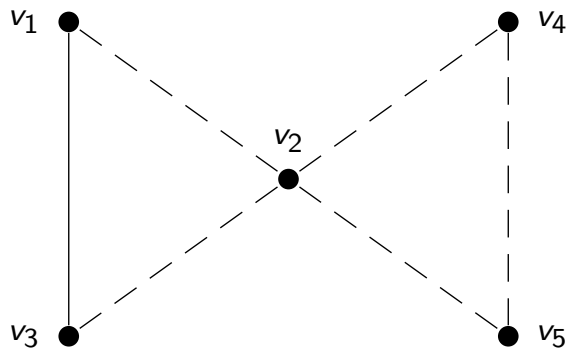
- In a CC, for every two x,y, it holds that $x \doteq^* y$ and $x \not\doteq^* y$.



- x, y, z, x with the (dis-)equalities $x \doteq y$, $y \doteq z$, $z \not\doteq x$ form a CC
- $x \doteq^* y$ because of x, y and $x \doteq y \in E_{\doteq}$
- $x \not\doteq^* y$ because of x, z, y and $z \not\doteq x \in E_{\not\doteq}$ and $y \doteq z \in E_{\doteq}$

Contradictory cycles and simple contradictory cycles

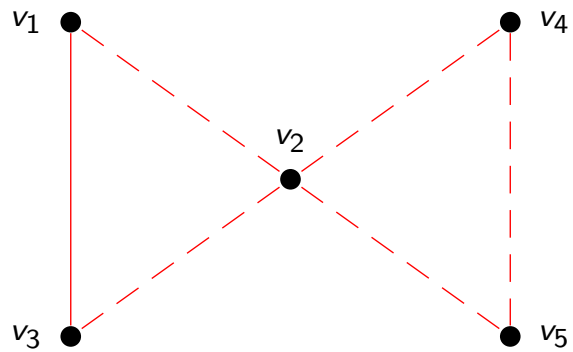
Contradictory cycle $v_1, v_2, v_4, v_5, v_2, v_3, v_1$



Contradictory cycle:	A cycle with exactly one disequality edge.
Simple cycle:	Only the start vertex is repeated.
Simple CC:	Both conditions are satisfied.

Contradictory cycles and simple contradictory cycles

Contradictory cycle $v_1, v_2, v_4, v_5, v_2, v_3, v_1$

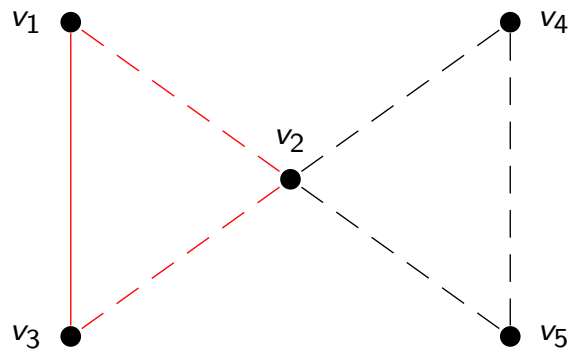


Theorem

Every contradictory cycle is either simple or contains a simple contradictory cycle.

Contradictory cycles and simple contradictory cycles

Simple contradictory cycle v_1, v_2, v_3, v_1



Theorem

Every contradictory cycle is either simple or contains a simple contradictory cycle.

Subgraphs and satisfiability

Definition

A subgraph of $G^E(\varphi^E)$ is called E-satisfiable iff the conjunction of the predicates represented by its edges is E-satisfiable.

Theorem

A subgraph of $G^E(\varphi^E)$ is E-unsatisfiable iff it contains a contradictory cycle.

Example

- Consider the contradictory cycle (CC) $x \doteq y$, $y \doteq z$, $x \not\doteq z$.
- From $x \doteq y$ and $y \doteq z$ and transitivity

$$x \doteq y \wedge y \doteq z \rightarrow x \doteq z$$

we derive $x \doteq z$, contradicting $x \not\doteq z$ from the CC.

Outline

The simplification algorithm

Algorithm: SIMPLIFY-EQUALITY-FORMULA

Input: An equality formula φ^E in NNF.

Output: An equality formula ψ^E equi-satisfiable with φ^E, with length less than or equal to the length of φ^E.

1. Let $\psi^E := \varphi^E$.
2. Construct the equality graph $G^E(\psi^E)$.
3. Replace each pure literal in ψ^E whose corresponding edge is not part of a simple contradictory cycle with *true*.
4. Simplify ψ^E with respect to Boolean constants *true* and *false*.
5. If any rewriting has occurred in the previous two steps, go to 2.
6. Return ψ^E.

The simplification algorithm

Algorithm: SIMPLIFY-EQUALITY-FORMULA

Input: An equality formula φ^E in NNF.

Output: An equality formula ψ^E equi-satisfiable with φ^E, with length less than or equal to the length of φ^E.

1. Let $\psi^E := \varphi^E$.
2. Construct the equality graph $G^E(\psi^E)$.
3. Replace each pure literal in ψ^E whose corresponding edge is not part of a simple contradictory cycle with *true*.
4. Simplify ψ^E with respect to Boolean constants *true* and *false*.
5. If any rewriting has occurred in the previous two steps, go to 2.
6. Return ψ^E.

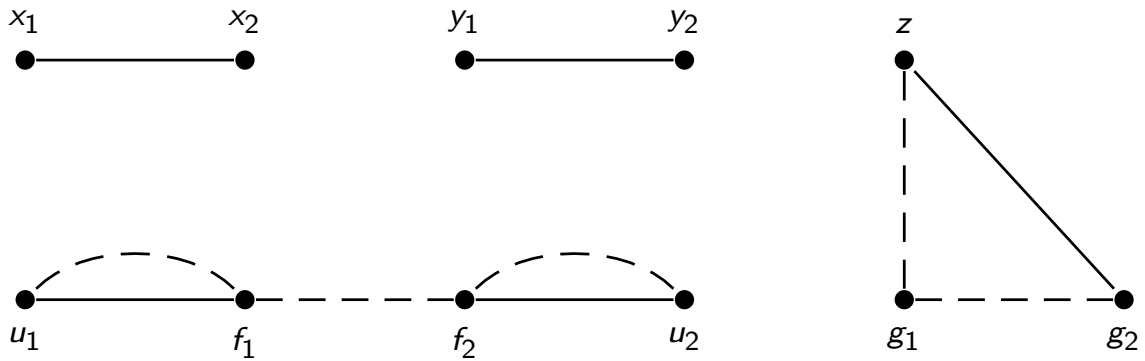
A literal ℓ is a pure literal in a formula λ if ℓ occurs either positively or negatively in λ.

Simplifications: The formula and the graph

$$\varphi^E: \quad ((x_1 \mathrel{\dot{\neq}} x_2 \vee y_1 \mathrel{\dot{\neq}} y_2 \vee f_1 \doteq f_2) \wedge (u_1 \mathrel{\dot{\neq}} f_1 \vee u_2 \mathrel{\dot{\neq}} f_2 \vee g_1 \doteq g_2) \wedge (u_1 \doteq f_1 \vee u_2 \doteq f_2 \vee z \doteq g_1)) \wedge z \mathrel{\dot{\neq}} g_2$$

$$E_{\doteq}: \quad \{f_1 \doteq f_2, g_1 \doteq g_2, u_1 \doteq f_1, u_2 \doteq f_2, z \doteq g_1\}$$

$$E_{\dot{\neq}}: \quad \{x_1 \mathrel{\dot{\neq}} x_2, y_1 \mathrel{\dot{\neq}} y_2, u_1 \mathrel{\dot{\neq}} f_1, u_2 \mathrel{\dot{\neq}} f_2, z \mathrel{\dot{\neq}} g_2\}$$



The edges (x_1, x_2), (y_1, y_2) and (f_1, f_2) are not part of any simple contradictory cycle $\rightsquigarrow$ set corresponding literals true

Simplifying the formula

$$\psi^E: \quad ((x_1 \mathrel{\dot{\neq}} x_2 \vee y_1 \mathrel{\dot{\neq}} y_2 \vee f_1 \doteq f_2) \wedge (u_1 \mathrel{\dot{\neq}} f_1 \vee u_2 \mathrel{\dot{\neq}} f_2 \vee g_1 \doteq g_2) \wedge (u_1 \doteq f_1 \vee u_2 \doteq f_2 \vee z \doteq g_1)) \wedge z \mathrel{\dot{\neq}} g_2$$

Simplifying the formula

$$\psi_1{}^E: \quad \big((\mathit{true} \vee \mathit{true} \vee \mathit{true}) \wedge (u_1 \mathrel{\dot{\neq}} f_1 \vee u_2 \mathrel{\dot{\neq}} f_2 \vee g_1 \doteq g_2) \wedge (u_1 \doteq f_1 \vee u_2 \doteq f_2 \vee z \doteq g_1)\big) \wedge z \mathrel{\dot{\neq}} g_2$$

Simplifying the formula

$$\psi_1{}^E: \quad ((\mathit{true} \vee \mathit{true} \vee \mathit{true}) \wedge (u_1 \mathrel{\dot{\neq}} f_1 \vee u_2 \mathrel{\dot{\neq}} f_2 \vee g_1 \doteq g_2) \wedge (u_1 \doteq f_1 \vee u_2 \doteq f_2 \vee z \doteq g_1)) \wedge z \mathrel{\dot{\neq}} g_2$$

Propositional simplifications result in:

$$\psi_2^E: \quad ((u_1 \mathrel{\dot{\neq}} f_1 \vee u_2 \mathrel{\dot{\neq}} f_2 \vee g_1 \doteq g_2) \wedge (u_1 \doteq f_1 \vee u_2 \doteq f_2 \vee z \doteq g_1)) \wedge z \mathrel{\dot{\neq}} g_2$$

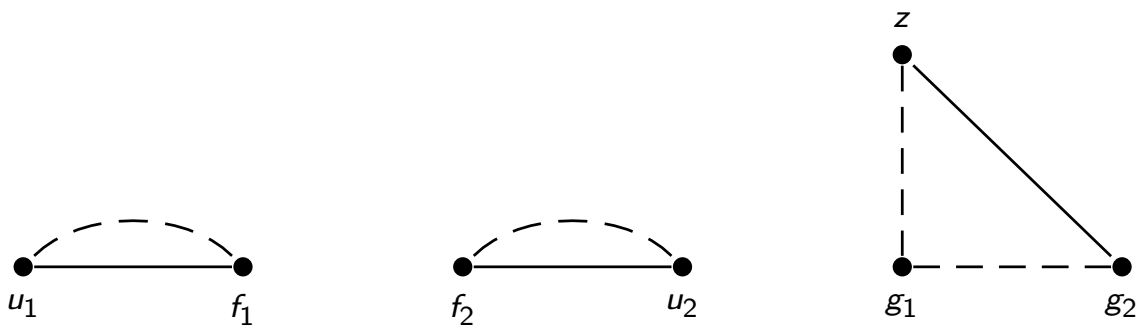
φ^E and ψ_2^E are equi-satisfiable, but ψ_2^E is smaller than φ^E.

Simplifying the formula

After the first round of simplification, we have

$$\psi_2^E: \quad ((u_1 \mathrel{\dot{\neq}} f_1 \vee u_2 \mathrel{\dot{\neq}} f_2 \vee g_1 \doteq g_2) \wedge (u_1 \doteq f_1 \vee u_2 \doteq f_2 \vee z \doteq g_1)) \wedge z \mathrel{\dot{\neq}} g_2$$

We go back to step 2 in the algorithm and construct $G^E(\psi_2^E)$:



No more simplifications are possible. ${\psi_2}^E$ is returned.

Outline

Reduction of E-formulas to propositional formulas

The basic idea

- The described method is called the sparse method.
- For φ^E, generate two formulas $e(\varphi^E)$ and B_t such that

$$\varphi^E \text{ is } E\text{-satisfiable} \quad \text{iff} \quad e(\varphi^E) \wedge B_t \text{ is satisfiable}$$

- $e(\varphi^E)$ is the propositional skeleton generated as follows:
  - Choose an ordering on variables and constants.
  - Orient the equations according to the ordering.
  - Replace $x_i \doteq x_j$ by $e_{i,j}$.
- To maintain equi-satisfiability, add transitivity constraints. (B_t is a conjunction of such constraints).

The construction of a propositional skeleton: An example

Let φ^E be $x_1 \doteq x_2 \wedge ((x_2 \doteq x_3 \wedge x_3 \not\doteq x_1) \vee x_2 \not\doteq x_1)$.

- Fix an ordering, e.g., $x_1 < x_2 < x_3$.
- Orient the equations: $x_3 \doteq x_1 \mapsto x_1 \doteq x_3$, $\quad x_2 \doteq x_1 \mapsto x_1 \doteq x_2$
- Then $e(\varphi^E)$ is $e_{1,2} \wedge ((e_{2,3} \wedge \neg e_{1,3}) \vee \neg e_{1,2})$
- It holds that if φ^E is satisfiable then so is $e(\varphi^E)$
- The converse is false, because ...
- $e(\varphi^E)$ is satisfiable but φ^E is E-unsatisfiable (☞ next slide)

φ^E and its propositional skeleton

- Reconsider φ^E: $x_1 \doteq x_2 \wedge ((x_2 \doteq x_3 \wedge x_3 \dot{\neq} x_1) \vee x_2 \dot{\neq} x_1)$.
- Then $e(\varphi^E)$ is $e_{1,2} \wedge ((e_{2,3} \wedge \neg e_{1,3}) \vee \neg e_{1,2})$.
- $e(\varphi^E)$ is satisfiable.
  - Simply set $e_{1,2}$ and $e_{2,3}$ to true and $e_{1,3}$ to false.
- φ^E is E-unsatisfiable. Let us try to satisfy φ^E:
  - Then any E-interpretation must make $x_1 \doteq x_2$ and $x_2 \doteq x_3 \wedge x_1 \dot{\neq} x_3$ true.
  - But then transitivity results that $x_1 \doteq x_3$ has to be true.
  - This however contradicts that $x_1 \dot{\neq} x_3$ is true.

↪ Transitivity is important to maintain (un-)satisfiability!

Non-polar equality graphs for the construction of B_t

The construction of B_t is guided by cycles in the non-polar equality graph of φ^E.

Definition

The non-polar equality graph corresponding to φ^E, $G^E_{NP}(\varphi^E)$, is an undirected graph (V, E), where the nodes in V correspond to the variables in φ^E and the edges correspond to $At(\varphi^E)$ (i.e., all equality predicates define the edges).

☞ $G^E_{NP}(\varphi^E)$ is a degenerated version of $G^E(\varphi^E)$, because the polarity of predicates (i.e., the negation sign) is neglected.

The transitivity constraints in B_t

- For each cycle in $G^E_{NP}(\varphi^E)$ with n vertices, we add a conjunction of n implications of the form $p_1 \wedge \ldots \wedge p_{n-1} \rightarrow p_n$.

  ☞ The size of B_t depends on the number of cycles in the graph!

- Each implication forbids to set the "last" edge false if all the other edges have been set to true.

- Imposing this constraint for each of the edges in each one of the cycles is sufficient to obtain

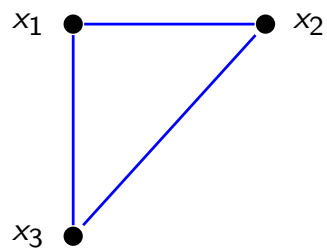
$$\varphi^E \text{ is } E\text{-satisfiable} \quad \text{iff} \quad e(\varphi^E) \wedge B_t \text{ is satisfiable}$$

Example: The formula φ^E, $G^E_{NP}(\varphi^E)$ and B_t

$$\varphi^E: \quad x_1 \doteq x_2 \wedge ((x_2 \doteq x_3 \wedge x_1 \not\doteq x_3) \vee x_1 \not\doteq x_2)$$

$$e(\varphi^E): \quad e_{1,2} \wedge ((e_{2,3} \wedge \neg e_{1,3}) \vee \neg e_{1,2})$$

$$G^E_{NP}(\varphi^E): \quad (\{x_1, x_2, x_3\}, \{(x_1, x_2), (x_2, x_3), (x_1, x_3)\})$$



- A (simple) cycle of length 3 in $G^E_{NP}(\varphi^E)$ is x_1, x_2, x_3, x_1 with corresponding edges (x_1, x_2), (x_2, x_3), (x_1, x_3) and corresponding variables $e_{1,2}$, $e_{2,3}$, $e_{1,3}$.
- Maintain equi-satisfiability by setting B_t to

$$(e_{1,2} \wedge e_{2,3} \rightarrow e_{1,3}) \wedge (e_{1,2} \wedge e_{1,3} \rightarrow e_{2,3}) \wedge (e_{2,3} \wedge e_{1,3} \rightarrow e_{1,2})$$

Is the use of simple cycles sufficient?

Theorem

In the construction of B_t, the use of simple cycles in $G^E_{NP}(\varphi^E)$ is sufficient to maintain equi-satisfiability.

Problem: Number of simple cycles in $G^E_{NP}(\varphi^E)$ can be exponential.

☞ Try to reduce the number of necessary cycles!

Solution: (Bryant and Velev): In the construction of B_t, it is sufficient to generate transitivity constraints from simple chord-free cycles (triangles) in the chordal version of the input graph $G^E_{NP}(\varphi^E)$.

Chords, chord-free cycles and chordal graphs

Definition

A chord of a cycle is an edge between two non-adjacent vertices of the cycle. If a cycle has no chords in a given graph, it is called a chord-free cycle.

Definition

An (undirected) graph G is called chordal (or triangulated) if no cycle of length ≥ 4 in G is chord-free.

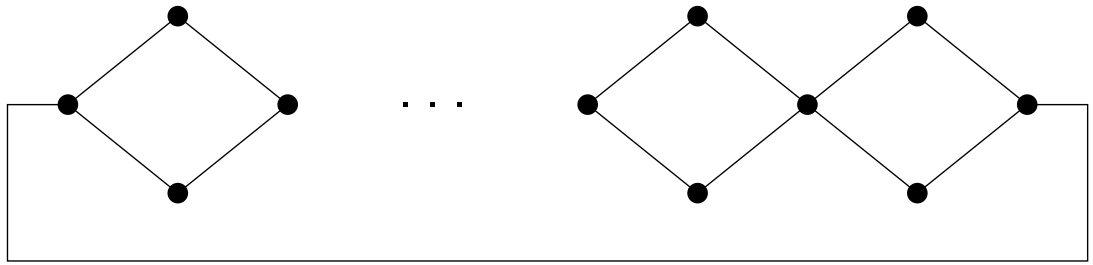
Q1 Do we really need chordal graphs?

Q2 If so, how do we construct them?

The motivation for chordal graphs

Observation

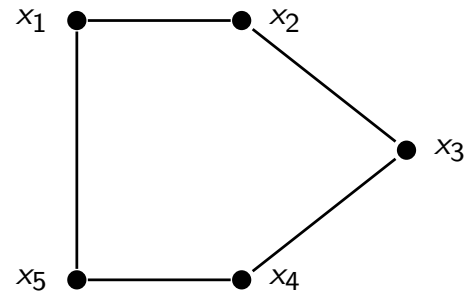
For arbitrary graphs G, the number of simple chord-free cycles in G can still be exponential in the number of vertices!



☞ We have to make the graph chordal (by adding edges)!

Example: Making a graph chordal

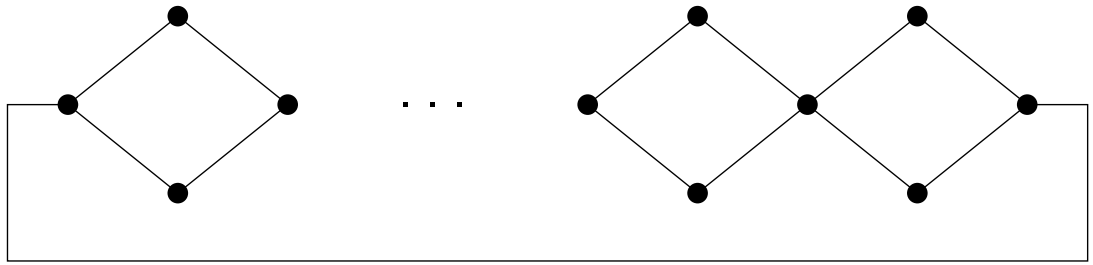
The simple cycle $x_1, x_2, x_3, x_4, x_5, x_1$ of length 5 is chord-free.



The motivation for chordal graphs

Observation

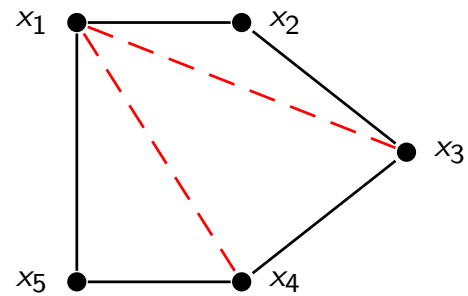
For arbitrary graphs G, the number of simple chord-free cycles in G can still be exponential in the number of vertices!



☞ We have to make the graph chordal (by adding edges)!

Example: Making a graph chordal

We can make the graph chordal by, e.g., introducing the red edges!



An algorithm for making a graph chordal

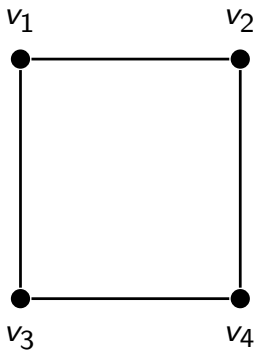
Any graph can be made chordal (in PTIME) by adding edges!

Procedure: Let an undirected graph $G = (V, E)$ be given.

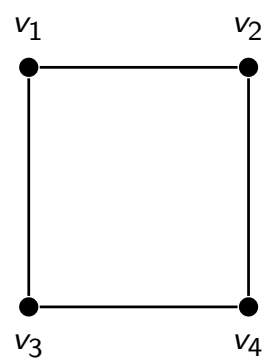
1. While $V \neq \{\}$ do
   - 1.1 Select a vertex v from V
   - 1.2 Add $E'(v) := \{(u, w) \mid (u, v), (v, w) \in E, (u, w) \notin E\}$ to E
   - 1.3 Remove v and all edges containing v from G
2. Return original G extended by all edges added in 1.2

☞ Result depends on the elimination ordering for vertices

v_1, v_2, v_3, v_4



v_2, v_1, v_3, v_4



An algorithm for making a graph chordal

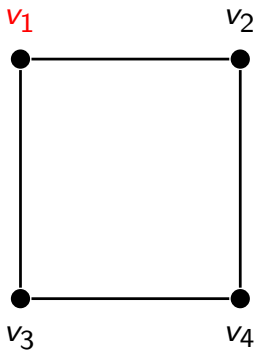
Any graph can be made chordal (in PTIME) by adding edges!

Procedure: Let an undirected graph $G = (V, E)$ be given.

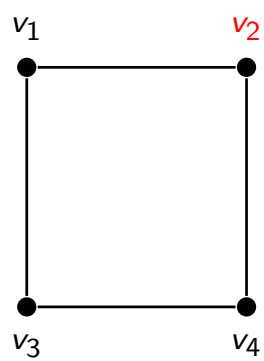
1. While $V \neq \{\}$ do
   - 1.1 Select a vertex v from V
   - 1.2 Add $E'(v) := \{(u, w) \mid (u, v), (v, w) \in E, (u, w) \notin E\}$ to E
   - 1.3 Remove v and all edges containing v from G
2. Return original G extended by all edges added in 1.2

☞ Result depends on the elimination ordering for vertices

v_1, v_2, v_3, v_4



v_2, v_1, v_3, v_4



An algorithm for making a graph chordal

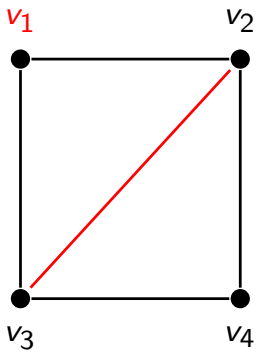
Any graph can be made chordal (in PTIME) by adding edges!

Procedure: Let an undirected graph $G = (V, E)$ be given.

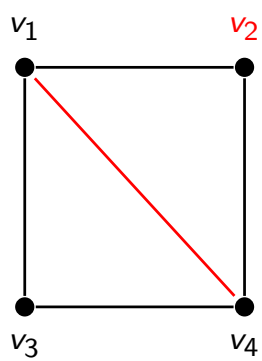
1. While $V \neq \{\}$ do
   - 1.1 Select a vertex v from V
   - 1.2 Add $E'(v) := \{(u, w) \mid (u, v), (v, w) \in E, (u, w) \notin E\}$ to E
   - 1.3 Remove v and all edges containing v from G
2. Return original G extended by all edges added in 1.2

☞ Result depends on the elimination ordering for vertices

v_1, v_2, v_3, v_4



v_2, v_1, v_3, v_4



An algorithm for making a graph chordal

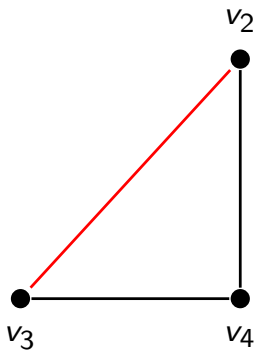
Any graph can be made chordal (in PTIME) by adding edges!

Procedure: Let an undirected graph $G = (V, E)$ be given.

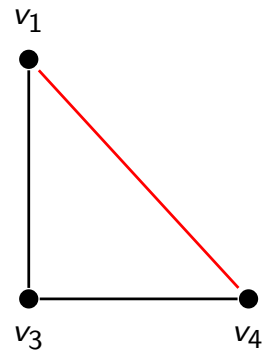
1. While $V \neq \{\}$ do
   - 1.1 Select a vertex v from V
   - 1.2 Add $E'(v) := \{(u, w) \mid (u, v), (v, w) \in E, (u, w) \notin E\}$ to E
   - 1.3 Remove v and all edges containing v from G
2. Return original G extended by all edges added in 1.2

☞ Result depends on the elimination ordering for vertices

$v_2,\ v_3,\ v_4$



$v_1,\ v_3,\ v_4$



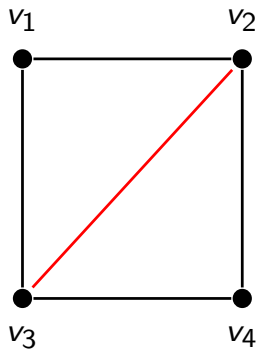
An algorithm for making a graph chordal

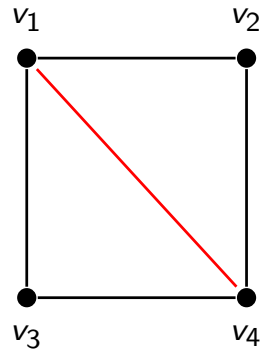
Any graph can be made chordal (in PTIME) by adding edges!

Procedure: Let an undirected graph $G = (V, E)$ be given.

1. While $V \neq \{\}$ do
   - 1.1 Select a vertex v from V
   - 1.2 Add $E'(v) := \{(u, w) \mid (u, v), (v, w) \in E, (u, w) \notin E\}$ to E
   - 1.3 Remove v and all edges containing v from G
2. Return original G extended by all edges added in 1.2

☞ Result depends on the elimination ordering for vertices





The two alternative chordal graphs are given above.

What do we win with chordal graphs?

- We have argued that using cycles in non-chordal graphs $G^E_{NP}(\varphi^E)$ makes B_t exponential.
- The number of simple chord-free cycles (triangles) is polynomial (cubic) in the size of $G^E_{NP}(\varphi^E)$.
- We know that triangles are sufficient for the generation of transitivity constraints (in B_t).
- Consequently, the size of B_t is polynomial in the size of φ^E (because the size of $G^E_{NP}(\varphi^E)$ is polynomial in the size of φ^E).
- B_t may have new variables (from newly introduced edges).

The reduction algorithm

From equality logic to propositional logic

Input: An already simplified equality formula φ^E (in NNF)
Output: A propositional formula equi-satisfiable with φ^E

1. Construct a Boolean formula $e(\varphi^E)$ by replacing any $x_i \doteq x_j$ in φ^E by a Boolean variable $e_{i,j}$.
2. Construct the non-polar equality graph $G^E_{NP}(\varphi^E)$.
3. Make $G^E_{NP}(\varphi^E)$ chordal.
4. Set B_t to true.
5. For each triangle $(e_{i,j}, e_{j,k}, e_{k,i})$ in $G^E_{NP}(\varphi^E)$ do

$$\begin{aligned} B_t \quad := \quad & (e_{i,j} \wedge e_{j,k} \rightarrow e_{i,k}) \wedge \\ & (e_{i,j} \wedge e_{i,k} \rightarrow e_{j,k}) \wedge \\ & (e_{i,k} \wedge e_{j,k} \rightarrow e_{i,j}) \wedge B_t \end{aligned}$$

6. Return $e(\varphi^E) \wedge B_t$.

The reduction algorithm

From equality logic to propositional logic

Input: An already simplified equality formula φ^E (in NNF)
Output: A propositional formula equi-satisfiable with φ^E

1. Construct a Boolean formula $e(\varphi^E)$ by replacing any $x_i \doteq x_j$ in φ^E by a Boolean variable e
2. Construct the no… raph $G^E_{NP}(\varphi^E)$.
3. Make G^E_{NP}(…
4. Set B…
5. …$_{i,j}, e_{j,k}, e_{k,i})$ in $G^E_{NP}(\varphi^E)$ do

$$\begin{aligned} B_t \quad := \quad & (e_{i,j} \wedge e_{j,k} \rightarrow e_{i,k}) \wedge \\ & (e_{i,j} \wedge e_{i,k} \rightarrow e_{j,k}) \wedge \\ & (e_{i,k} \wedge e_{j,k} \rightarrow e_{i,j}) \wedge B_t \end{aligned}$$

6. Return $e(\varphi^E) \wedge B_t$.

Improvement

Use $G^E(\varphi^E)$ for the analysis and obtain less transitivity constraints!

Literature

Different kinds of reduction have been developed

1. R. Bryant, M. Velev: Boolean Satisfiability with Transitivity Constraints. CAV 2000, 85-98
2. R. Bryant, M. Velev: Boolean satisfiability with transitivity constraints. ACM Trans. Comput. Log., 3(4):604-627. 2002
3. O. Meir, O. Strichman: Yet Another Decision Procedure for Equality Logic. CAV 2005, 307-320
4. M. Rozanov, O. Strichman: Generating Minimum Transitivity Constraints in P-time for Deciding Equality Logic. Electr. Notes Theor. Comput. Sci. 198(2):3-17. 2008

Learning objectives

You should be able to

- explain the concepts of equality graphs, non-polar equality graphs and chordal graphs and their use in the reduction,
- construct such graphs from given E-formulas,
- identify different kind of cycles in graphs,
- simplify formulas,
- perform correctness proofs for the simplification methods,
- make a graph chordal,
- apply the reduction.