Decision Procedures in First Order Logic

Decision Procedures for Equality Logic

Outline

- √ Introduction
  - √ Definition, complexity
  - √ Reducing Uninterpreted Functions to Equality Logic
  - √ Using Uninterpreted Functions in proofs
  - √ Simplifications
- Introduction to the decision procedures
  - The framework: assumptions and Normal Forms
  - General terms and notions
  - Solving a conjunction of equalities
  - Simplifications

Basic assumptions and notations

- Input formulas are in NNF
- Input formulas are checked for satisfiability

- Formula with Uninterpreted Functions: ϕ^{UF}
- Equality formula: ϕ^{E}

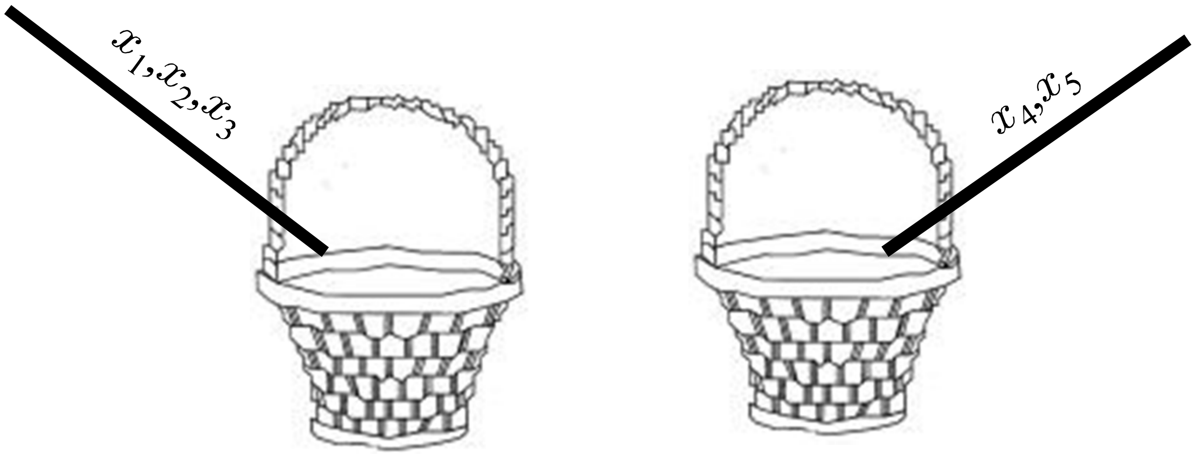
First: conjunction of equalities

- Input: A conjunction of equalities and disequalities

1. Define an equivalence class for each variable. For each equality $x = y$ unite the equivalence classes of x and y. Repeat until convergence.
2. For each disequality $u \neq v$ if u is in the same equivalence class as v return 'UNSAT'.
3. Return 'SAT'.

Example

- $x_1 = x_2$ Æ $x_2 = x_3$ Æ $x_4 = x_5$ Æ $x_5 \neq x_1$

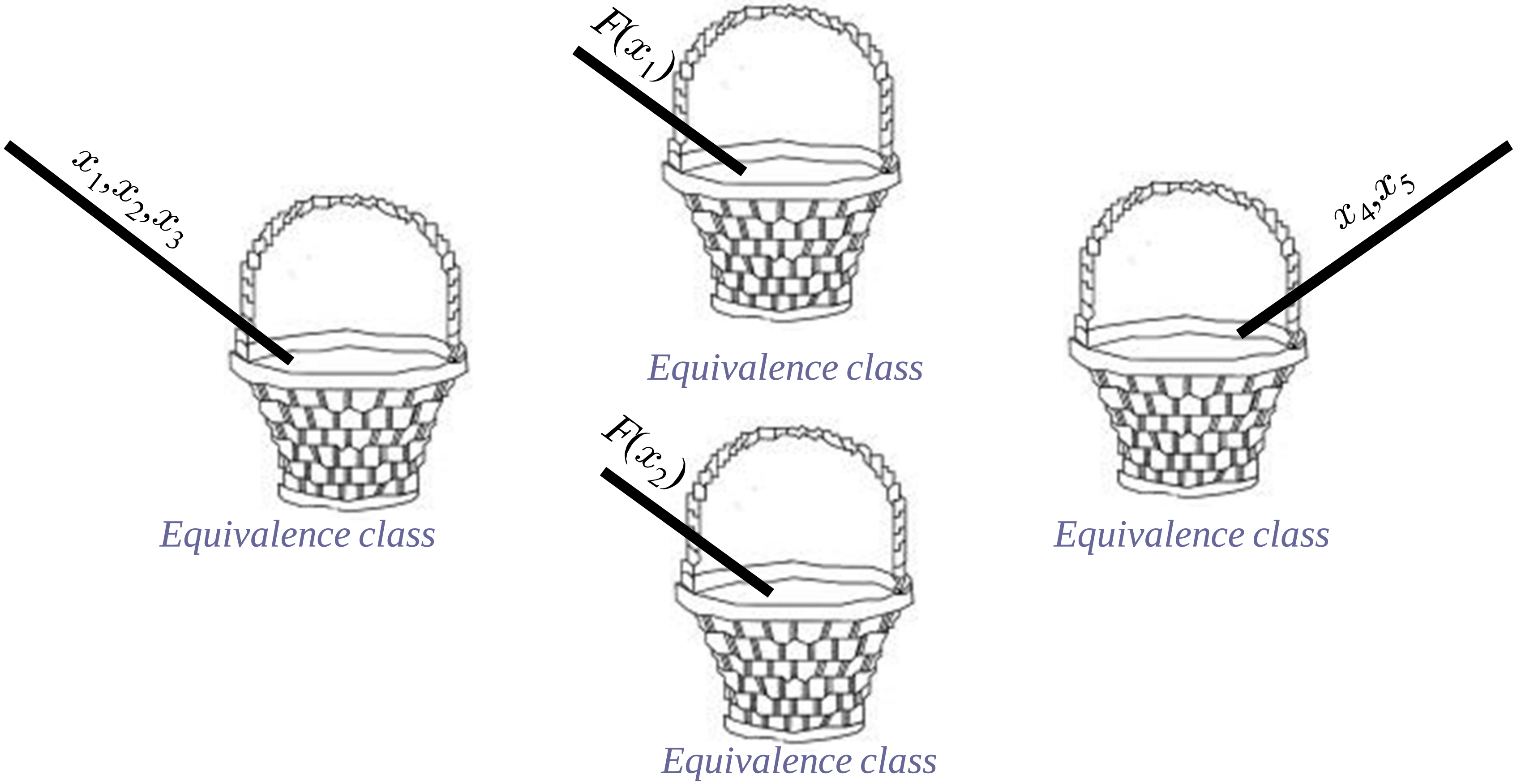


Equivalence class *Equivalence class*

Is there a disequality between members of the same class ?

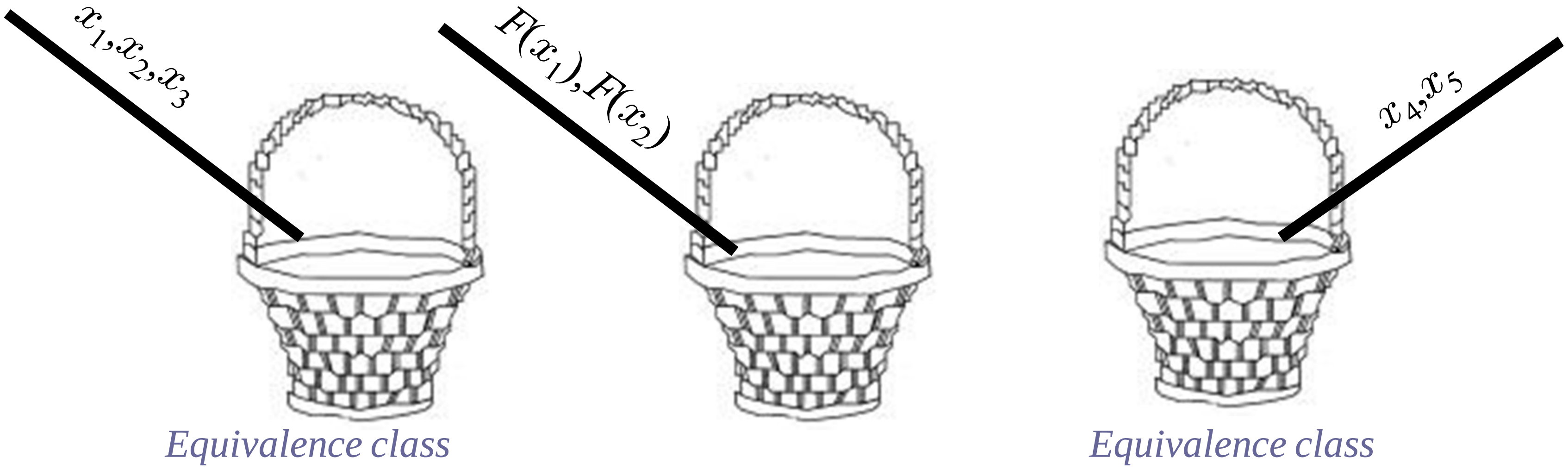
Next: add Uninterpreted Functions

- $x_1 = x_2$ Æ $x_2 = x_3$ Æ $x_4 = x_5$ Æ $x_5 \neq x_1$ Æ $F(x_1) \neq F(x_2)$



Next: Compute the *Congruence Closure*

- $x_1 = x_2$ Æ $x_2 = x_3$ Æ $x_4=x_5$ Æ $x_5 \neq x_1$ Æ $F(x_1)\neq F(x_2)$

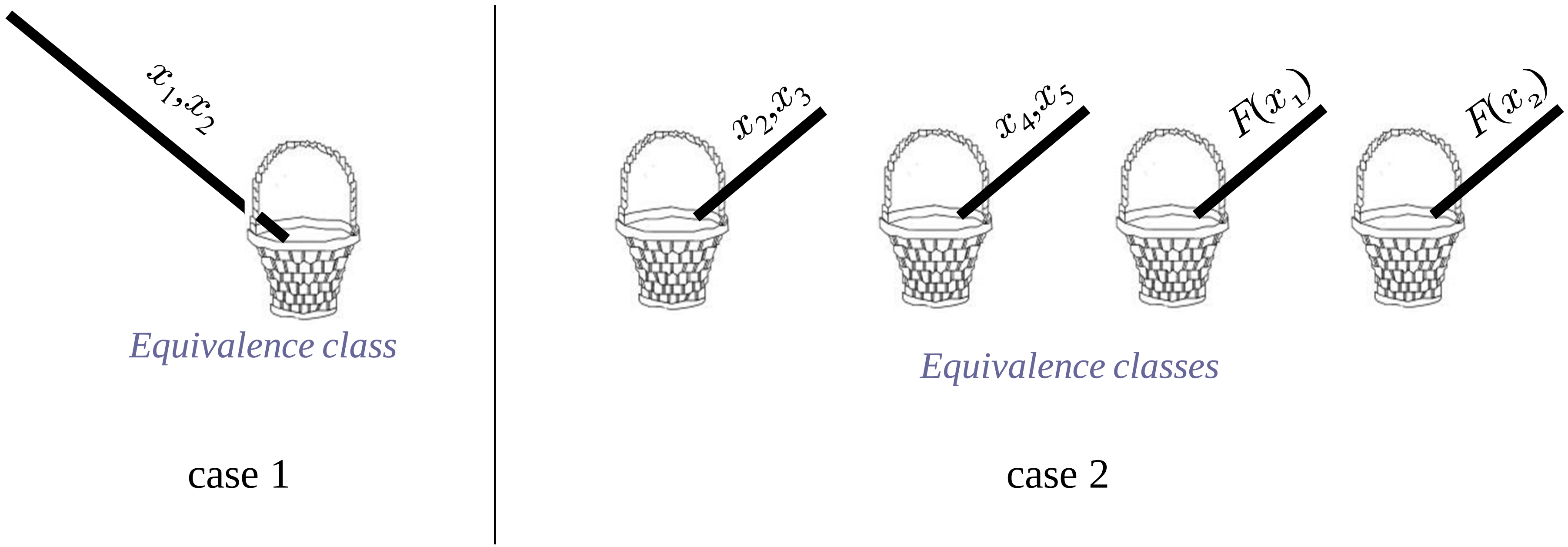


Now - is there a disequality between members of the same class ?

This is called the Congruence Closure

And now: consider a Boolean structure

- $x_1 = x_2$ Ç ($x_2 = x_3$ Æ $x_4 = x_5$ Æ $x_5 \neq x_1$ Æ $F(x_1) \neq F(x_2)$)



Syntactic case splitting: this is what we want to avoid!

Deciding Equality Logic with UFs

- Input: Equality Logic formula ϕ^{UF}
- Convert ϕ^{UF} to DNF
- For each clause:
  - Define an equivalence class for each variable and each function instance.
  - For each equality $x = y$ unite the equivalence classes of x and y. For each function symbol F, unite the classes of $F(x)$ and $F(y)$. Repeat until convergence.
  - If all disequalities are between terms from different equivalence classes, return 'SAT'.
- Return 'UNSAT'.

Algorithm 3.3.1: ACKERMANN'S-REDUCTION

Input: An EUF formula φ^{UF} with m instances of an uninterpreted function F $\boxed{m}$

Output: An equality logic formula φ^{E} such that φ^{E} is valid if and only if φ^{UF} is valid

1. Assign indices to the uninterpreted-function instances from subexpressions outwards. Denote by F_i the instance of F that is given the index i, and by $arg(F_i)$ its single argument. $\boxed{F_i}$ $\boxed{arg(F_i)}$
2. Let $flat^{\mathrm{E}} \doteq \mathcal{T}(\varphi^{\mathrm{UF}})$, where $\mathcal{T}$ is a function that takes an EUF formula (or term) as input and transforms it to an equality formula (or term, respectively) by replacing each uninterpreted-function instance F_i with a new term-variable f_i (in the case of nested functions, only the variable corresponding to the most external instance remains). $\boxed{flat^{\mathrm{E}}}$ $\boxed{\mathcal{T}}$
3. Let FC^{E} denote the following conjunction of functional consistency constraints: $\boxed{FC^{\mathrm{E}}}$

$$FC^{\mathrm{E}} := \bigwedge_{i=1}^{m-1} \bigwedge_{j=i+1}^{m} (\mathcal{T}(arg(F_i)) = \mathcal{T}(arg(F_j))) \implies f_i = f_j \; .$$

4. Let

$$\varphi^{\mathrm{E}} := FC^{\mathrm{E}} \implies flat^{\mathrm{E}} \; .$$

Return φ^{E}.

Algorithm 3.3.2: BRYANT'S-REDUCTION

Input: An EUF formula φ^{UF} with m instances of an uninterpreted function F

Output: An equality logic formula φ^{E} such that φ^{E} is valid if and only if φ^{UF} is valid

1. Assign indices to the uninterpreted-function instances from subexpressions outwards. Denote by F_i the instance of F that is given the index i, and by $arg(F_i)$ its single argument.
2. Let $flat^{\mathrm{E}} = \mathcal{T}^\star(\varphi^{\mathrm{UF}})$, where $\mathcal{T}^\star$ is a function that takes an EUF formula (or term) as input and transforms it to an equality formula (or term, respectively) by replacing each uninterpreted-function instance F_i with a new term-variable $F_i^\star$ (in the case of nested functions, only the variable corresponding to the most external instance remains).
3. For $i \in \{1, \ldots, m\}$, let f_i be a new variable, and let $F_i^\star$ be defined as follows:

$$F_i^\star := \left(\begin{array}{lll} \text{case } \mathcal{T}^\star(arg(F_1^\star)) & = \mathcal{T}^\star(arg(F_i^\star)) & : f_1 \\ \quad\vdots & \quad\vdots & \\ \mathcal{T}^\star(arg(F_{i-1}^\star)) & = \mathcal{T}^\star(arg(F_i^\star)) & : f_{i-1} \\ \text{TRUE} & & : f_i \end{array} \right) . \tag{3.19}$$

Finally, let

$$FC^{\mathrm{E}} := \bigwedge_{i=1}^{m} F_i^\star . \tag{3.20}$$

4. Let

$$\varphi^{\mathrm{E}} := FC^{\mathrm{E}} \implies flat^{\mathrm{E}} . \tag{3.21}$$

Return φ^{E}.

$\boxed{flat^{\mathrm{E}}}$ $\boxed{\mathcal{T}^\star}$ $\boxed{F_i^\star}$

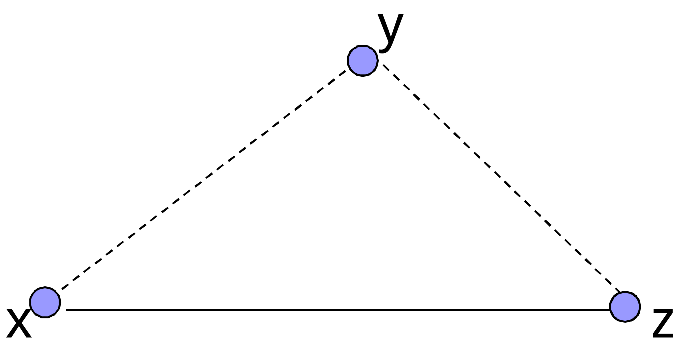
Basic notions

$$\phi^E: \quad x = y \wedge y = z \wedge z \neq x$$

- The Equality predicates: $\{x = y, y = z, z \neq x\}$ which we can break to two sets:

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

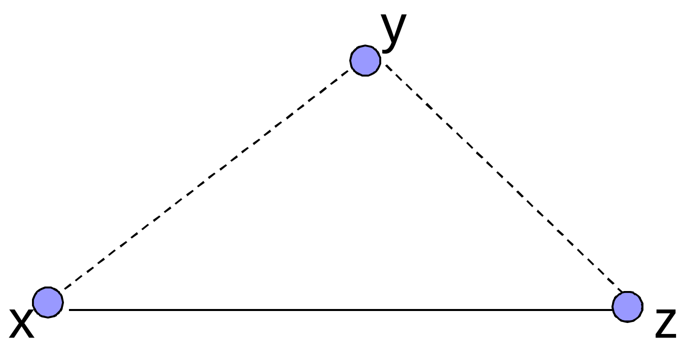
- The Equality Graph $G^E(\phi^E) = \langle V, E_=, E_{\neq} \rangle$ (a.k.a "E-graph")



Basic notions

ϕ_1^E: $x = y \wedge y = z \wedge z \neq x$ *unsatisfiable*

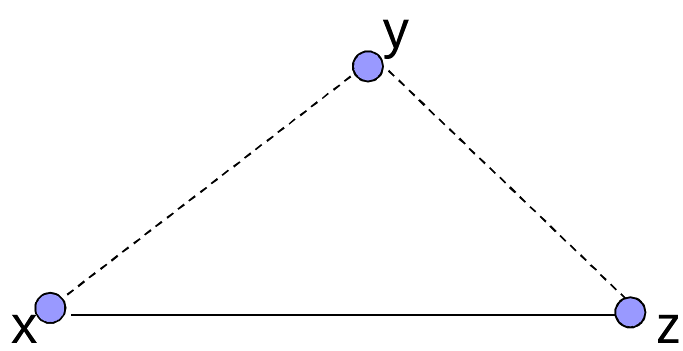
ϕ_2^E: $x = y \wedge y = z \vee z \neq x$ *satisfiable*



The graph $G^E(\phi^E)$ represents an abstraction of ϕ^E

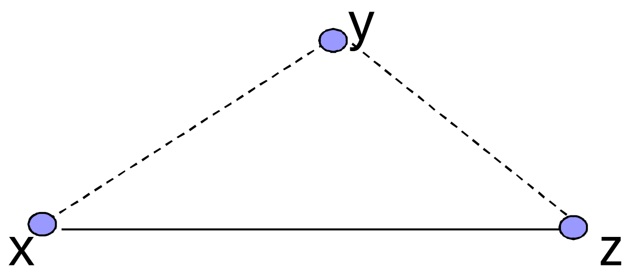
It ignores the Boolean structure of ϕ^E

Basic notions



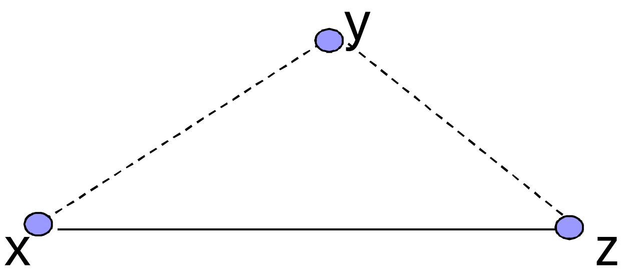
- *Dfn:* a path made of $E_=$ edges is an *Equality Path*. we write $x =^* z$.
- *Dfn:* a path made of $E_=$ edges + exactly one edge from $E_{\neq}$ is a *Disequality Path*. We write $x \neq^* y$.

Basic notions



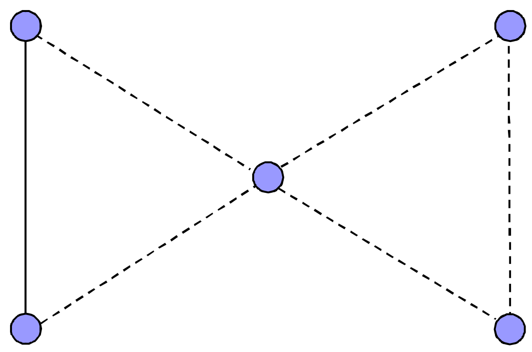
- Dfn. *A cycle with one disequality edge is a* Contradictory Cycle.
- In a Contradictory Cycle, for every two nodes x,y it holds that $x =^* y$ and $x \neq^* y$.

Basic notions



- Dfn: *A subgraph is called satisfiable iff the conjunction of the predicates represented by its edges is satisfiable.*
- Thm: *A subgraph is unsatisfiable iff it contains a Contradictory cycle*

Basic notions



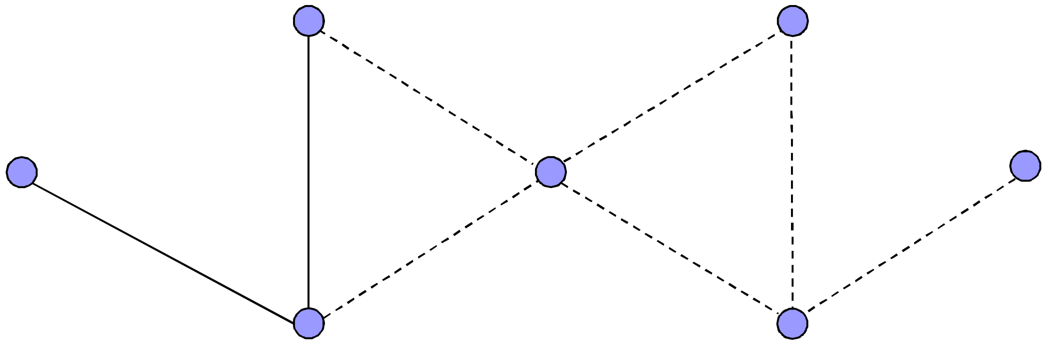
- *Thm: Every Contradictory Cycle is either simple or contains a simple contradictory cycle*

Algorithm 4.3.1: SIMPLIFY-EQUALITY-FORMULA

Input: An equality formula φ^{E}
Output: An equality formula $\varphi^{\mathrm{E}\prime}$ equisatisfiable with φ^{E}, with length less than or equal to the length of φ^{E}

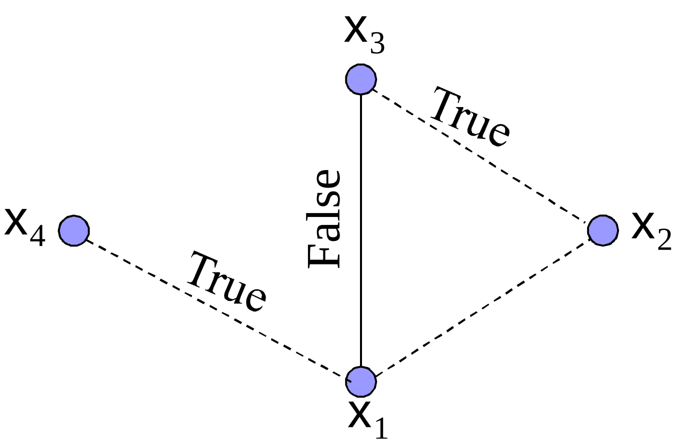
1. Let $\varphi^{\mathrm{E}\prime} := \varphi^{\mathrm{E}}$.
2. Construct the equality graph $G^{\mathrm{E}}(\varphi^{\mathrm{E}\prime})$.
3. Replace each pure literal in $\varphi^{\mathrm{E}\prime}$ whose corresponding edge is not part of a simple contradictory cycle with TRUE.
4. Simplify $\varphi^{\mathrm{E}\prime}$ with respect to the Boolean constants TRUE and FALSE (e.g., replace TRUE $\vee\, \phi$ with TRUE, and FALSE $\wedge\, \phi$ with FALSE).
5. If any rewriting has occurred in the previous two steps, go to step 2.
6. Return $\varphi^{\mathrm{E}\prime}$.

Simplifications, again



- Let S be the set of edges that are not part of any Contradictory Cycle
- Thm: *replacing all solid edges in* S *with False, and all dashed edges in* S *with True, preserves satisfiability*

Simplification: example



- ($x_1 = x_2$ Ç $x_1 = x_4$) Æ
  ($x_1 \neq x_3$ Ç $x_2 = x_3$)
- ~~($x_1 = x_2$ Ç True) Æ~~
  ($x_1 \neq x_3$ Ç $x_2 = x_3$)
- (: False Ç True) = True
- Satisfiable!

Syntactic vs. Semantic splits

- So far we saw how to handle disjunctions through syntactic case-splitting.
- There are much better ways to do it than simply transforming it to DNF:
  - Semantic Tableaux,
  - SAT-based splitting,
  - others…
- We will investigate some of these methods later in the course.

Syntactic vs. Semantic splits

- Now we start looking at methods that split the search space instead. This is called *semantic splitting*.

- SAT is a very good engine for performing semantic splitting, due to its ability to guide the search, prune the search-space etc.