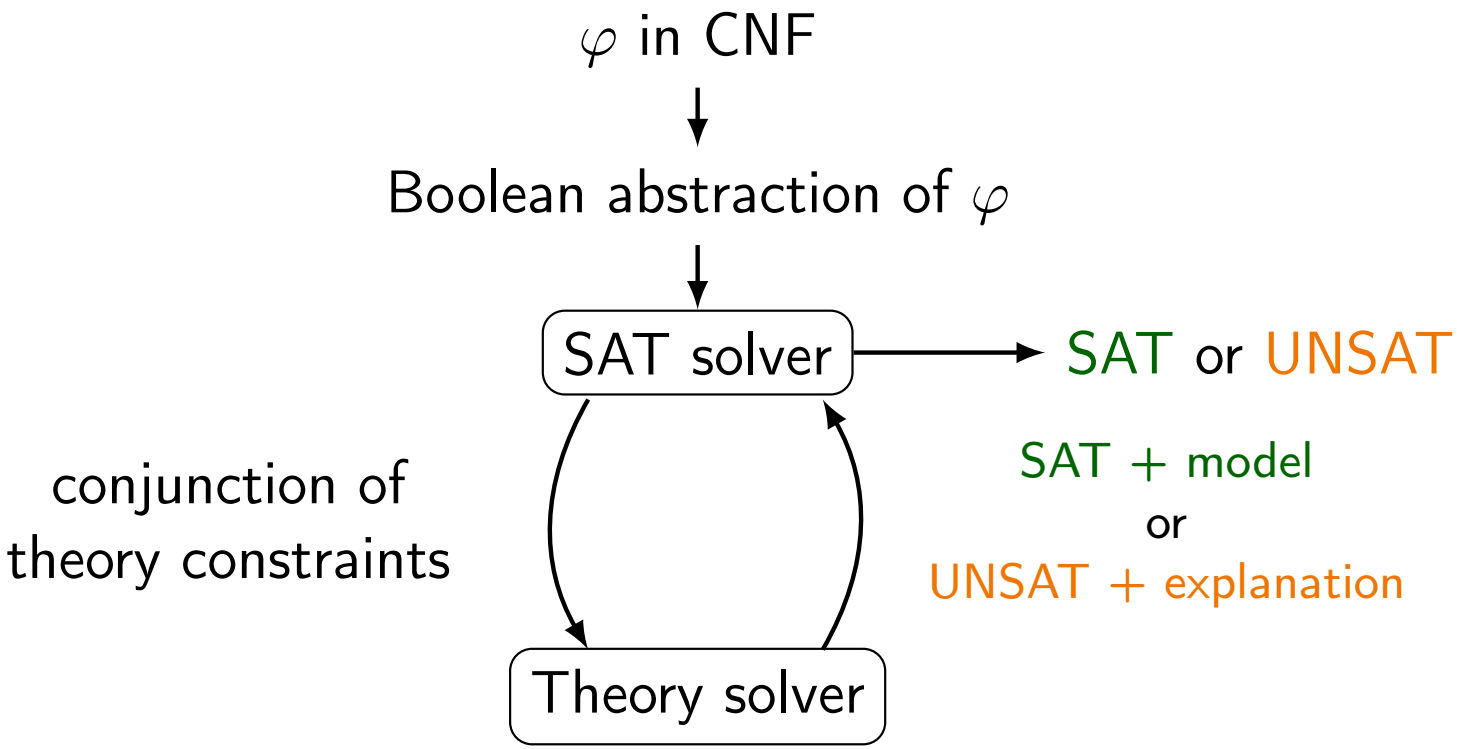
Practical Course: SMT Solving

Preprocessing, Union Find and Theory Propagation

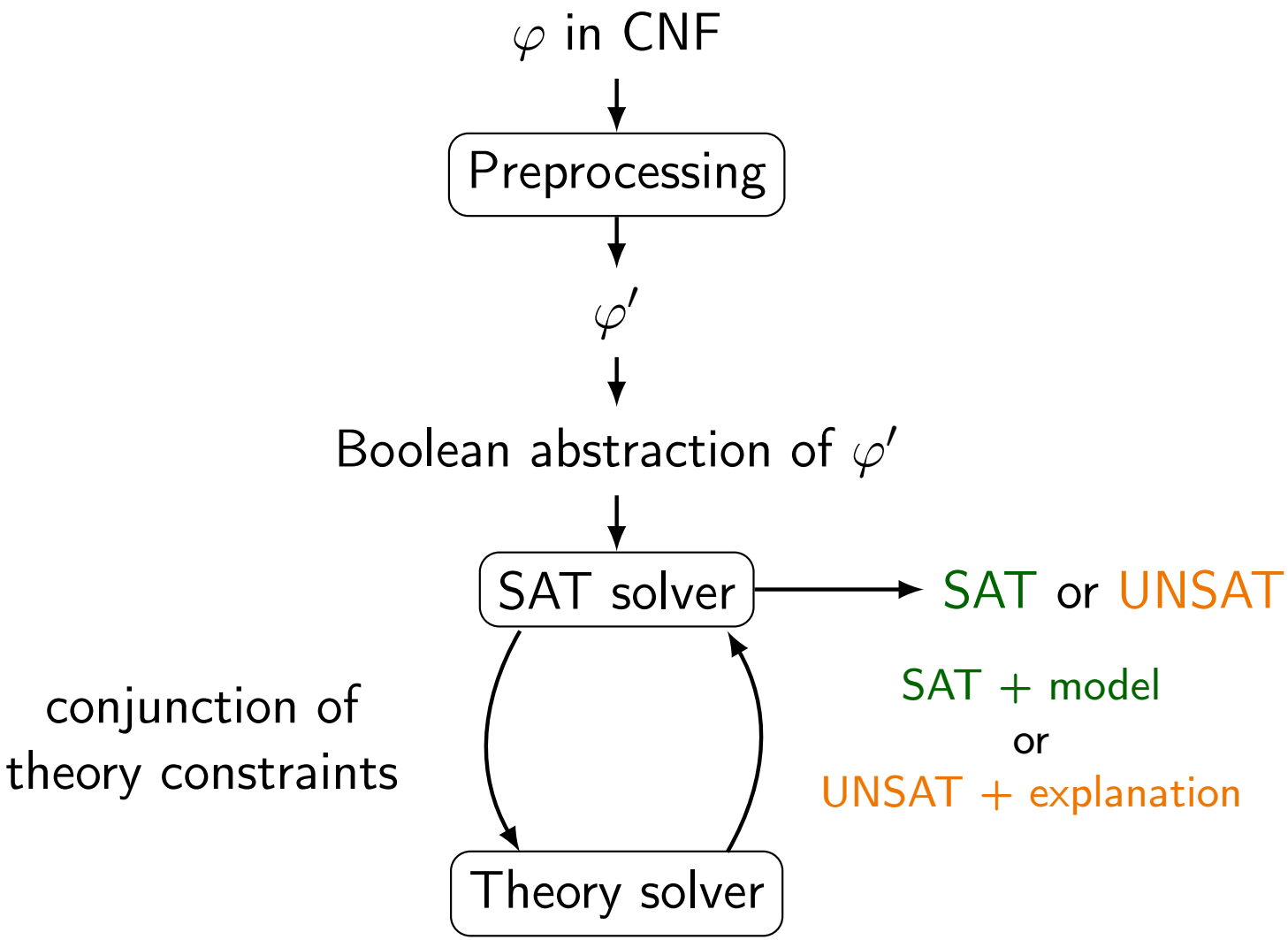
Erika Ábrahám, Rebecca Haehn, Gereon Kremer, Stefan Schupp

Winter term 2018/19

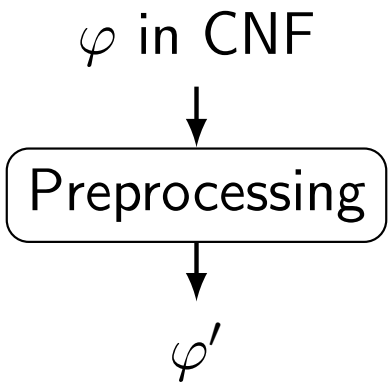
Preprocessing



Preprocessing



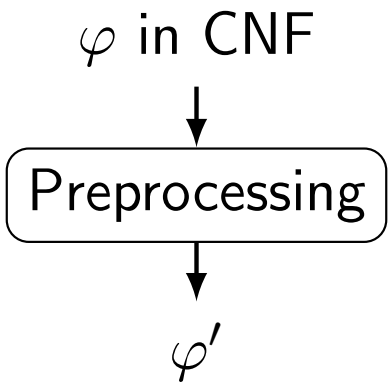
Preprocessing



- equivalence preserving simplifications: $\varphi \equiv \varphi'$, e.g.

$$f(a) = b \ \wedge \ g(d, f(a)) = c \ \wedge \ h(f(a), f(a), f(a)) = e$$
$$\equiv f(a) = b \ \wedge \ g(d, b) = c \ \wedge \ h(b, b, b) = e$$

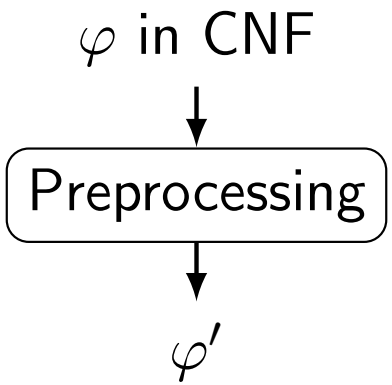
Preprocessing



- equivalence preserving simplifications: $\varphi \equiv \varphi'$, e.g.

$$f(a) = b \ \wedge \ g(d, f(a)) = c \ \wedge \ h(f(a), f(a), f(a)) = e$$
$$\equiv f(a) = b \ \wedge \ g(d, b) = c \ \wedge \ h(b, b, b) = e$$
$$\text{or } f(a) = f(a) \ \vee \ b = c \ \equiv \ \mathit{true}$$

Preprocessing



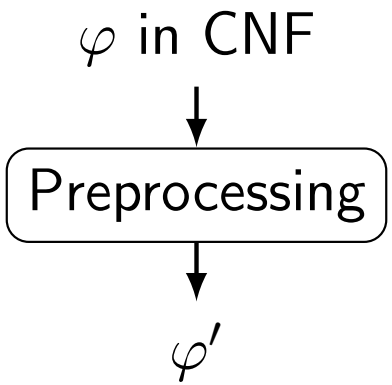
- equivalence preserving simplifications: $\varphi \equiv \varphi'$, e.g.

$$f(a) = b \ \wedge \ g(d, f(a)) = c \ \wedge \ h(f(a), f(a), f(a)) = e$$
$$\equiv f(a) = b \ \wedge \ g(d, b) = c \ \wedge \ h(b, b, b) = e$$
$$\text{or } f(a) = f(a) \ \vee \ b = c \ \equiv \ \mathit{true}$$

- φ and φ' may be only equisatisfiable, e.g.

$$f(a) = b \ \wedge \ g(d, f(a)) = c \ \wedge \ h(f(a), f(a), f(a)) = e$$
$$\rightarrow g(d, b) = c \ \wedge \ h(b, b, b) = e$$

Preprocessing



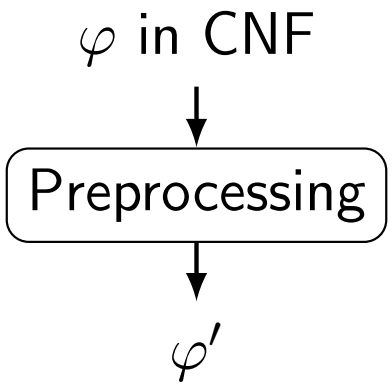
- equivalence preserving simplifications: $\varphi \equiv \varphi'$, e.g.

$$f(a) = b \ \wedge \ g(d, f(a)) = c \ \wedge \ h(f(a), f(a), f(a)) = e$$
$$\equiv f(a) = b \ \wedge \ g(d, b) = c \ \wedge \ h(b, b, b) = e$$
$$\text{or } f(a) = f(a) \ \vee \ b = c \ \equiv \ \mathit{true}$$

- φ and φ' may be only equisatisfiable, e.g.

$$f(a) = b \ \wedge \ g(d, f(a)) = c \ \wedge \ h(f(a), f(a), f(a)) = e$$
$$\rightarrow g(d, b) = c \ \wedge \ h(b, b, b) = e$$

Preprocessing



- equivalence preserving simplifications: $\varphi \equiv \varphi'$, e.g.

$$f(a) = b \ \wedge \ g(d, f(a)) = c \ \wedge \ h(f(a), f(a), f(a)) = e$$
$$\equiv f(a) = b \ \wedge \ g(d, b) = c \ \wedge \ h(b, b, b) = e$$
$$\text{or } f(a) = f(a) \ \vee \ b = c \ \equiv \ \mathit{true}$$

- φ and φ' may be only equisatisfiable, e.g.

$$f(a) = b \ \wedge \ g(d, f(a)) = c \ \wedge \ h(f(a), f(a), f(a)) = e$$
$$\rightarrow g(d, b) = c \ \wedge \ h(b, b, b) = e$$

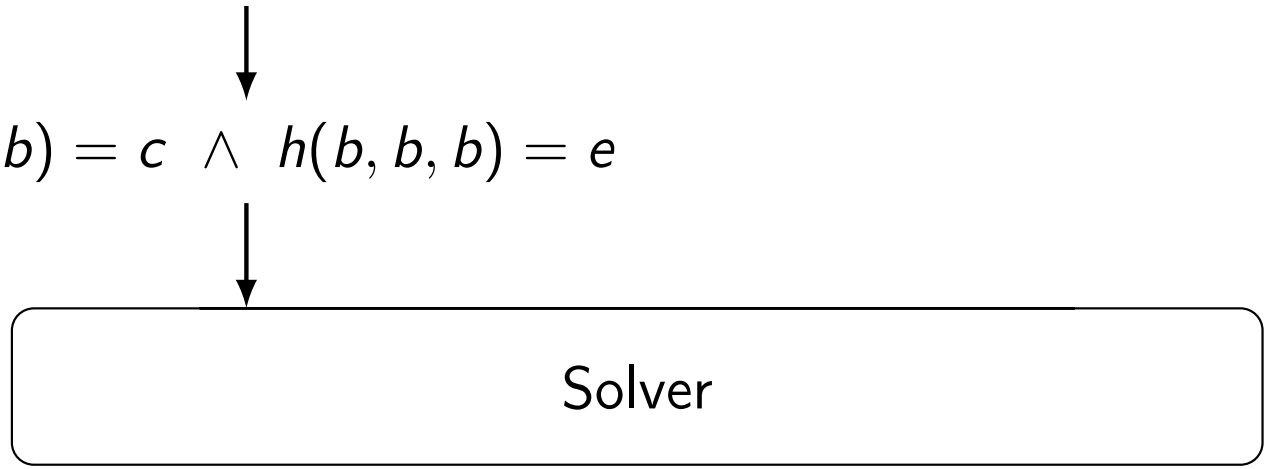
Equisatisfiability

- φ and φ' may be only equisatisfiable

Equisatisfiability

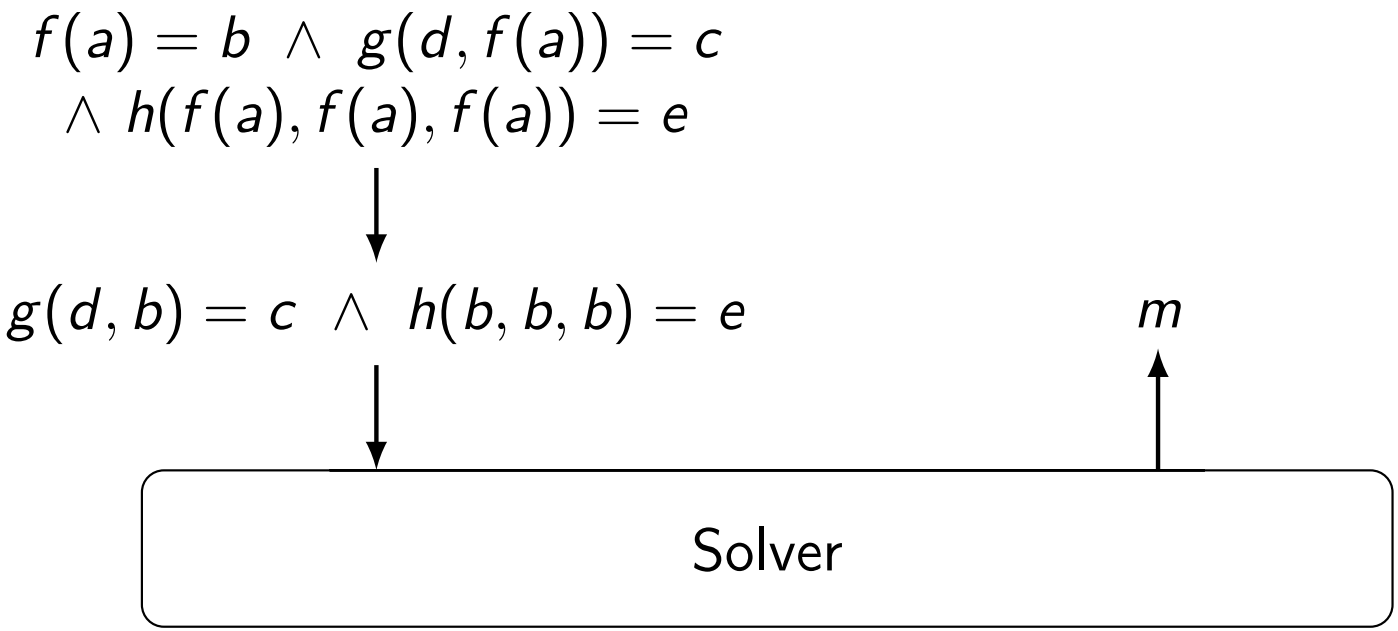
- φ and φ' may be only equisatisfiable
- models and proofs need to be converted

$$f(a) = b \ \wedge \ g(d, f(a)) = c$$
$$\wedge \ h(f(a), f(a), f(a)) = e$$



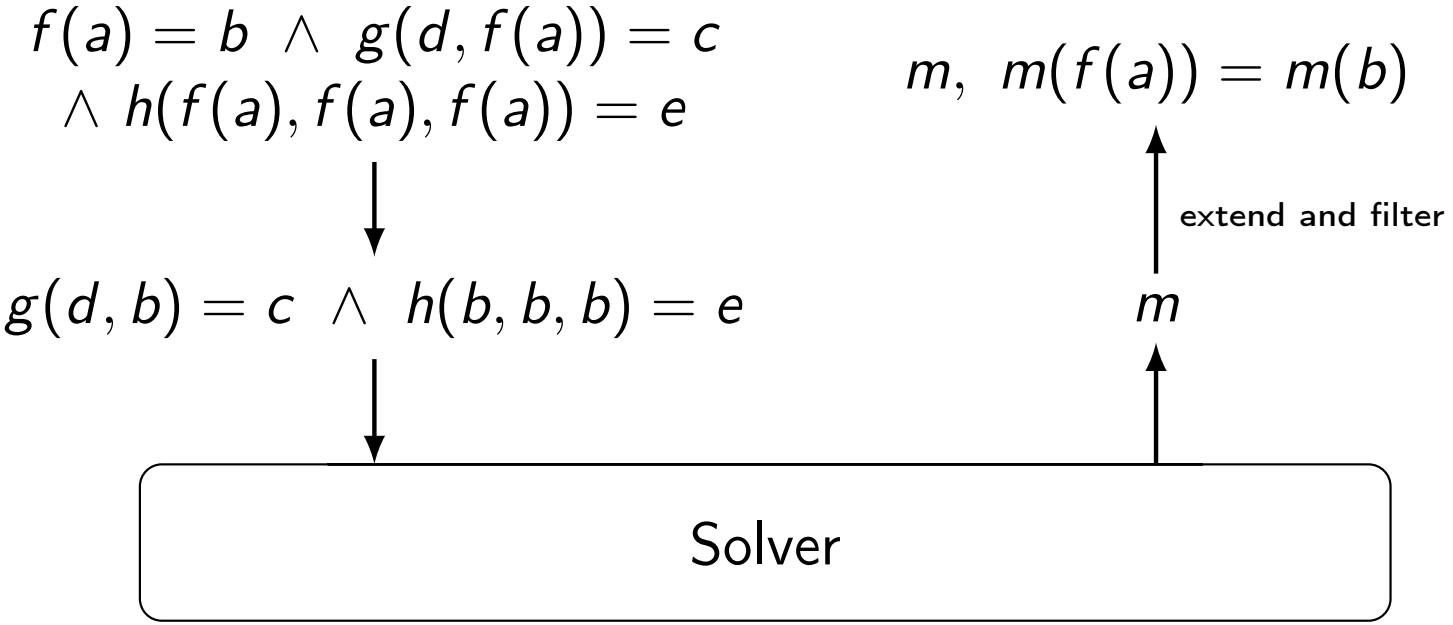
Equisatisfiability

- φ and φ' may be only equisatisfiable
- models and proofs need to be converted



Equisatisfiability

- φ and φ' may be only equisatisfiable
- models and proofs need to be converted



Term Removal

Let φ be a QF_UF formula in CNF.

Term Removal

Let φ be a QF_UF formula in CNF.

- $t = t \ \wedge \ \varphi \ \mapsto \ \varphi$

Term Removal

Let φ be a QF_UF formula in CNF.

- $t = t \ \wedge \ \varphi \ \mapsto \ \varphi$
- $x = y \ \wedge \ \varphi \ \mapsto \ \varphi[x/y]$ if x, y are variables

Term Removal

Let φ be a QF_UF formula in CNF.

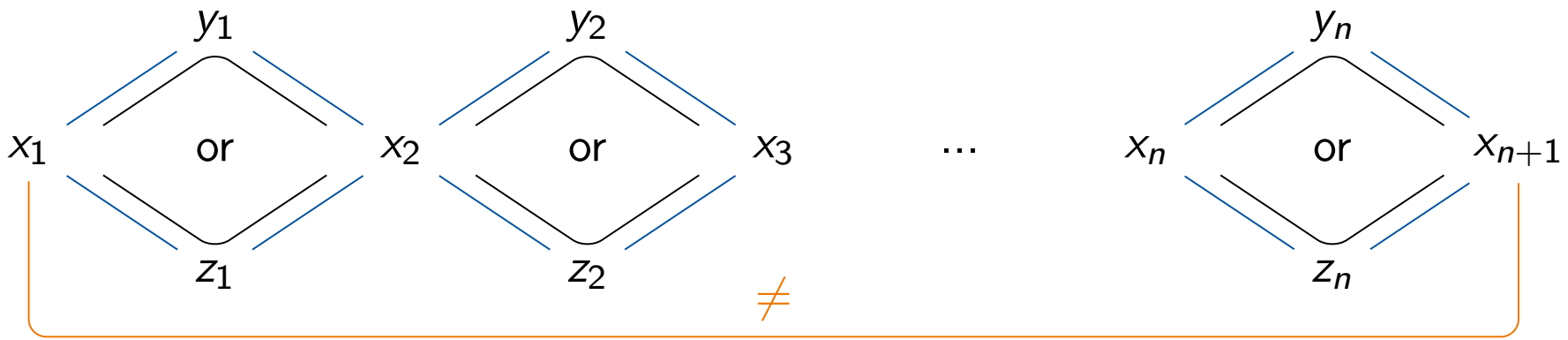
- $t = t \ \wedge \ \varphi \ \mapsto \ \varphi$
- $x = y \ \wedge \ \varphi \ \mapsto \ \varphi[x/y]$ if x, y are variables
- $t_1 = t_2 \ \wedge \ \varphi \ \mapsto \ t_1 = t_2 \ \wedge \ \varphi[t_1/t_2]$ if t_1, t_2 are terms, t_1 should not be a subterm of t_2

Term Removal

Let φ be a QF_UF formula in CNF.

- $t = t \ \wedge \ \varphi \ \mapsto \ \varphi$
- $x = y \ \wedge \ \varphi \ \mapsto \ \varphi[x/y]$ if x, y are variables
- $t_1 = t_2 \ \wedge \ \varphi \ \mapsto \ t_1 = t_2 \ \wedge \ \varphi[t_1/t_2]$ if t_1, t_2 are terms, t_1 should not be a subterm of t_2
- $\varphi \ \mapsto \ \varphi[t/x]$ if t is a function of terms, for each subterm s of t occurs no literal $s = u$ (for any term u) in φ and x is a new variable

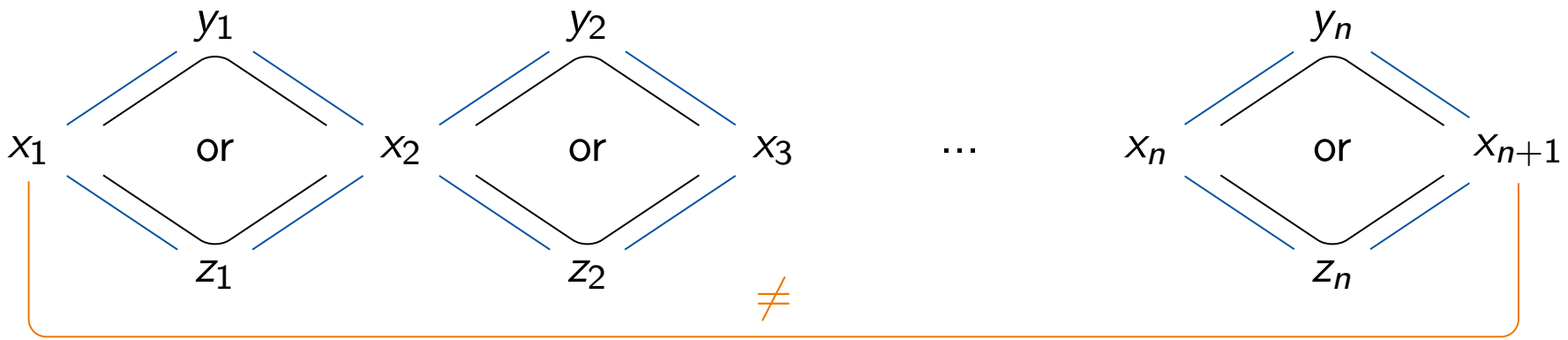
Diamond Chains



$$(x_1 = y_1 = x_2 \vee x_1 = z_1 = x_2) \wedge (x_2 = y_2 = x_3 \vee x_2 = z_2 = x_3) \wedge \cdots \wedge (x_n = y_n = x_{n+1} \vee x_n = z_n = x_{n+1})$$

$$\wedge\ x_1 \neq x_{n+1}$$

Diamond Chains



$$(x_1 = y_1 = x_2 \vee x_1 = z_1 = x_2) \wedge (x_2 = y_2 = x_3 \vee x_2 = z_2 = x_3) \wedge \ldots \wedge (x_n = y_n = x_{n+1} \vee x_n = z_n = x_{n+1}) \wedge x_1 \neq x_{n+1}$$

Can be simplified to *false*. (But how?)

Term Simplification

Currify:

- use only one function symbol f ('apply')

Term Simplification

Currify:

- use only one function symbol f ('apply')
- e.g. $g(a) \mapsto f(g, a)$ and
  $h(a, g(b), c) \mapsto f(f(f(h, a), f(g, b)), c)$

Term Simplification

Currify:

- use only one function symbol f ('apply')
- e.g. $g(a) \mapsto f(g, a)$ and
  $h(a, g(b), c) \mapsto f(f(f(h, a), f(g, b)), c)$
- linear growth in size

Term Simplification

Currify:

- use only one function symbol f ('apply')
- e.g. $g(a) \mapsto f(g, a)$ and $h(a, g(b), c) \mapsto f(f(f(h, a), f(g, b)), c)$
- linear growth in size

Flatten:

Term Simplification

Currify:

- use only one function symbol f ('apply')
- e.g. $g(a) \mapsto f(g, a)$ and $h(a, g(b), c) \mapsto f(f(f(h, a), f(g, b)), c)$
- linear growth in size

Flatten:

- replace subterms with new variables

Term Simplification

Currify:

- use only one function symbol f ('apply')
- e.g. $g(a) \mapsto f(g, a)$ and
  $h(a, g(b), c) \mapsto f(f(f(h, a), f(g, b)), c)$
- linear growth in size

Flatten:

- replace subterms with new variables
- e.g. $f(f(f(h, a), f(g, b)), c)$
  $\mapsto f(h, a) = d \wedge f(g, b) = e \wedge f(d, e)) = k \wedge f(k, c) = m$

Union Find [1]

- merge tree data structure F to store equivalence classes

[1] as in http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf

Union Find [1]

- merge tree data structure F to store equivalence classes
- $F(x)$ is the predecessor of x, if x is a root $F(x) = x$

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

- merge tree data structure F to store equivalence classes
- $F(x)$ is the predecessor of x, if x is a root $F(x) = x$
- initially $F(x) = x$ for all x

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

- merge tree data structure F to store equivalence classes
- $F(x)$ is the predecessor of x, if x is a root $F(x) = x$
- initially $F(x) = x$ for all x
- $F^*(x)$ is the root of the tree that contains x

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

- merge tree data structure F to store equivalence classes
- $F(x)$ is the predecessor of x, if x is a root $F(x) = x$
- initially $F(x) = x$ for all x
- $F^*(x)$ is the root of the tree that contains x
- x and y are equal if they are in the same tree (i.e. if $F^*(x) = F^*(y)$)

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

- merge tree data structure F to store equivalence classes
- $F(x)$ is the predecessor of x, if x is a root $F(x) = x$
- initially $F(x) = x$ for all x
- $F^*(x)$ is the root of the tree that contains x
- x and y are equal if they are in the same tree (i.e. if $F^*(x) = F^*(y)$)
- let $sz(F, x)$ be the size of the tree containing x then merging the equivalence classes of x and y is defined as

$$union(F, x, y) = \begin{cases} F, & \textit{if } F^*(x) = F^*(y) \\ F[F^*(x) \mapsto F^*(y)], & \textit{if } sz(F, x) < sz(F, y) \\ F[F^*(y) \mapsto F^*(x)], & \textit{else} \end{cases}$$

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

$$x_1 = x_2 \wedge \ x_1 = x_3 \wedge \ x_2 = x_3 \wedge \ x_2 \neq x_4 \wedge \ x_4 = x_5$$

$$F = \{x_1 \mapsto x_1,\ x_2 \mapsto x_2,\ x_3 \mapsto x_3,\ x_4 \mapsto x_4,\ x_5 \mapsto x_5\}$$

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

$x_1 = x_2 \wedge\ x_1 = x_3 \wedge\ x_2 = x_3 \wedge\ x_2 \neq x_4 \wedge\ x_4 = x_5$

$F = \{x_1 \mapsto x_1,\ x_2 \mapsto x_1,\ x_3 \mapsto x_3,\ x_4 \mapsto x_4,\ x_5 \mapsto x_5\}$

$\rightarrow$ merge classes of x_1 and x_2

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

$x_1 = x_2 \wedge \ x_1 = x_3 \wedge \ x_2 = x_3 \wedge \ x_2 \neq x_4 \wedge \ x_4 = x_5$

$F = \{x_1 \mapsto x_1, \ x_2 \mapsto x_1, \ x_3 \mapsto x_1, \ x_4 \mapsto x_4, \ x_5 \mapsto x_5\}$

$\rightarrow$ merge classes of x_1 and x_3

[1] as in http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf

Union Find [1]

$x_1 = x_2 \wedge \ x_1 = x_3 \wedge \ x_2 = x_3 \wedge \ x_2 \neq x_4 \wedge \ x_4 = x_5$

$F = \{x_1 \mapsto x_1,\ x_2 \mapsto x_1,\ x_3 \mapsto x_1,\ x_4 \mapsto x_4,\ x_5 \mapsto x_5\}$

$\rightarrow$ x_2 and x_3 are already in the same class

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

$x_1 = x_2 \wedge \ x_1 = x_3 \wedge \ x_2 = x_3 \wedge \ x_2 \neq x_4 \wedge \ x_4 = x_5$

$F = \{x_1 \mapsto x_1, \ x_2 \mapsto x_1, \ x_3 \mapsto x_1, \ x_4 \mapsto x_4, \ x_5 \mapsto x_5\}$

$\rightarrow$ x_2 and x_4 are in different classes

[1] as in http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf

Union Find [1]

$x_1 = x_2 \wedge\ x_1 = x_3 \wedge\ x_2 = x_3 \wedge\ x_2 \neq x_4 \wedge\ x_4 = x_5$

$F = \{x_1 \mapsto x_1,\ x_2 \mapsto x_1,\ x_3 \mapsto x_1,\ x_4 \mapsto x_4,\ x_5 \mapsto x_4\}$

$\rightarrow$ merge classes of x_4 and x_5; x_2 and x_4 are still in different classes

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

$x_1 = x_2 \wedge\ x_1 = x_3 \wedge\ x_2 = x_3 \wedge\ x_2 \neq x_4 \wedge\ x_4 = x_5$

$F = \{x_1 \mapsto x_1,\ x_2 \mapsto x_1,\ x_3 \mapsto x_1,\ x_4 \mapsto x_4,\ x_5 \mapsto x_4\}$

- optimizations possible (e.g. path compression: update F when computing $F^*(x)$)

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

$x_1 = x_2 \wedge\ x_1 = x_3 \wedge\ x_2 = x_3 \wedge\ x_2 \neq x_4 \wedge\ x_4 = x_5$

$F = \{x_1 \mapsto x_1,\ x_2 \mapsto x_1,\ x_3 \mapsto x_1,\ x_4 \mapsto x_4,\ x_5 \mapsto x_4\}$

- optimizations possible (e.g. path compression: update F when computing $F^*(x)$)
- \+ fast look-up whether two elements are in the same equivalence class

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Union Find [1]

$x_1 = x_2 \wedge\ x_1 = x_3 \wedge\ x_2 = x_3 \wedge\ x_2 \neq x_4 \wedge\ x_4 = x_5$

$F = \{x_1 \mapsto x_1,\ x_2 \mapsto x_1,\ x_3 \mapsto x_1,\ x_4 \mapsto x_4,\ x_5 \mapsto x_4\}$

- optimizations possible (e.g. path compression: update F when computing $F^*(x)$)
- \+ fast look-up whether two elements are in the same equivalence class
- − non-incremental, has to be recomputed after backtracking

[1] as in `http://fm.csl.sri.com/SSFT12/smt-euf-arithmetic.pdf`

Theory Propagation

- a theory solver can use its partial model to induce truth-values for some constraints

Theory Propagation

- a theory solver can use its partial model to induce truth-values for some constraints
- e.g. $a = b \ \wedge \ b = c \ \Rightarrow \ f(a) = f(c) \mapsto \mathit{true}$

Theory Propagation

- a theory solver can use its partial model to induce truth-values for some constraints
- e.g. $a = b \ \wedge \ b = c \ \Rightarrow \ f(a) = f(c) \mapsto \mathit{true}$
- detect this and assign the respective constraints in the SAT solver

Theory Propagation

- a theory solver can use its partial model to induce truth-values for some constraints
- e.g. $a = b \ \wedge \ b = c \ \Rightarrow \ f(a) = f(c) \mapsto \mathit{true}$
- detect this and assign the respective constraints in the SAT solver
- trade-off between costs of theory propagation and the Boolean search space reduction it provides

Theory Propagation

- a theory solver can use its partial model to induce truth-values for some constraints
- e.g. $a = b \ \wedge \ b = c \ \Rightarrow \ f(a) = f(c) \mapsto \mathit{true}$
- detect this and assign the respective constraints in the SAT solver
- trade-off between costs of theory propagation and the Boolean search space reduction it provides
- in practice: incomplete theory propagation